TRABAJO DE FIN DE GRADO

Implementación de Algoritmos de Visión por Computador en Plataforma Android

Autor:

ALEJANDRO RAMOS LÓPEZ

Grado en Ingeniería Informática

Tutor:

Fernando García Fernández

Director de proyecto:

Juan Carmona Fernández

Septiembre 2014

TÍTULO: Implementación de algoritmos de Visión por Computador en Plataforma Android.

AUTOR: Alejandro Ramos López

TUTOR: Fernando García Fernández

EL TRIBUNAL

PRESIDENTE: _____

VOCAL: _____

SECRETARIO: _____

Realizado el acto de defensa y lectura del Trabajo de Fin de Grado el día 1 de Octubre de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

PRESIDENTE SECRETARIO VOCAL

ÍNDICE GENERAL

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

RESUMEN

En plena era del teléfono inteligente, en inglés "smartphone", miles de empresas y particulares buscan nuevas ideas que desarrollar para estos dispositivos. Cada día, decenas de todo tipo de aplicaciones son publicadas en las tiendas de las diferentes plataformas móviles. Sin embargo, se pueden contar con los dedos las aplicaciones que tienen puntos en común con el proyecto sobre el que trata esta memoria.

En este proyecto se establece el tratamiento de la imagen a través de computador como la base principal de un conjunto de aplicaciones futuras que puedan ser de utilidad tanto en el campo de la investigación como en los campos institucionales y comerciales.

El procesamiento de la imagen a través de computador es ya una realidad patente en multitud de campos: investigaciones para mejorar la vida de las personas invidentes, vehículos que, en función de una serie de patrones, conducen, se detienen o aparcan automáticamente, sistemas de seguridad, etc. Sin embargo, es un campo tímidamente explorado en dispositivos tan comunes como los smartphones.

Las librerías OpenCV, desarrolladas por Intel, son pieza clave en la investigación y el desarrollo de nuevos programas que utilicen la visión por computador. El problema radica en que, gran parte de esta investigación, necesita de notables cantidades de dinero, puesto que los dispositivos que se utilizan en este campo, añadido a lo experimental del asunto, son caros y no están al alcance de un desarrollador autodidacta.

¿Qué ocurriría si se lograra, en un futuro a corto plazo, establecer estas tecnologías en dispositivos tan comunes hoy en día como lo son los teléfonos móviles?

AGRADECIMIENTOS

A mis padres por todo su apoyo, gracias al cual estoy hoy aquí.

A mi novia Patricia.

A toda mi familia y amigos, en especial a Juan Carmona, por su confianza y su apoyo.

ACRÓNIMOS

Acrónimo	Significado
CONETIC	Confederación Española de Empresas de Tecnologías de la Información, Comunicaciones y Electrónica.
ETSI	European Telecommunications Standards Institute
FPS	Frames Per Second (Imágenes por segundo)
ITS	Intelligent Transportation Systems
LSI	Laboratorio de Sistemas Inteligentes
NDK	Native Development Kit
OPENCV	Open Source Computer Vision
OS	Operating System
RISC	Reduced Instruction Set Computer
SDK	Software Development Kit
SLAM	Simultaneous Localization And Mapping
SO	Sistema Operativo
TFG	Trabajo de Fin de Grado
UC3M	Universidad Carlos III de Madrid
XML	Extensible Markup Language

Tabla 1 - Acrónimos

1. INTRODUCCIÓN

El proyecto descrito en esta memoria tiene como objetivo principal el desarrollo de una aplicación base para el Laboratorio de Sistemas Inteligentes (LSI) de la Universidad Carlos III de Madrid, implementando la visión por computador en teléfonos inteligentes con sistema operativo Android.

La base de esta aplicación, a su vez dividida en "subaplicaciones", es la utilización de las librerías de visión artificial OpenCV.

Al principio del proyecto, la idea inicial era la realización de una aplicación Android que llevara a cabo la detección de diferentes objetos en función de una serie de patrones. Dos fueron las razones principales para elegir el sistema operativo de Google: por un lado, la gratuidad en el desarrollo, y por otro, el conocimiento previo del alumno, gracias al desarrollo de alguna aplicación durante el curso del Grado en Ingeniería Informática. En el sub-apartado "Elección del entorno de desarrollo" se explican más profundamente las razones por las cuales se eligió Android.

Si bien las librerías OpenCV están en auge, aún son un campo exiguamente explorado en numerosos lenguajes y plataformas, por lo que en muchas ocasiones el alumno ha sufrido la carencia de información en relación a este entorno concreto (Android-Java). Además, se han de destacar las evidentes limitaciones que produce trabajar en un dispositivo relativamente limitado, como puede ser un smartphone.

Cabe destacar que este proyecto tiene también como objetivo establecer un punto de inicio, una base, para proyectos posteriores, relacionados con Android y Opencv, desarrollados para el LSI. En el apartado "Trabajos futuros" se mencionarán algunos ejemplos más detalladamente.

2. ESTADO DEL ARTE

2.1. Android

Android [1] es un sistema operativo libre, gratuito y multiplataforma. Basado en Linux, fue inicialmente desarrollado para teléfonos móviles, si bien en la actualidad se encuentra en multitud de dispositivos como: tablets, portátiles, netbooks, relojes, auriculares, etc.

Las aplicaciones para Android se ejecutan en la máquina virtual Dalvik, diseñada con el objetivo de reducir la cantidad de memoria que las aplicaciones necesitan y que permite ejecutar varias instancias de la máquina virtual simultáneamente, dejando al sistema operativo subyacente la gestión de la memoria, de los hilos y de los procesos (en el caso de Android, el aislamiento de procesos es conocido como Sandboxing).

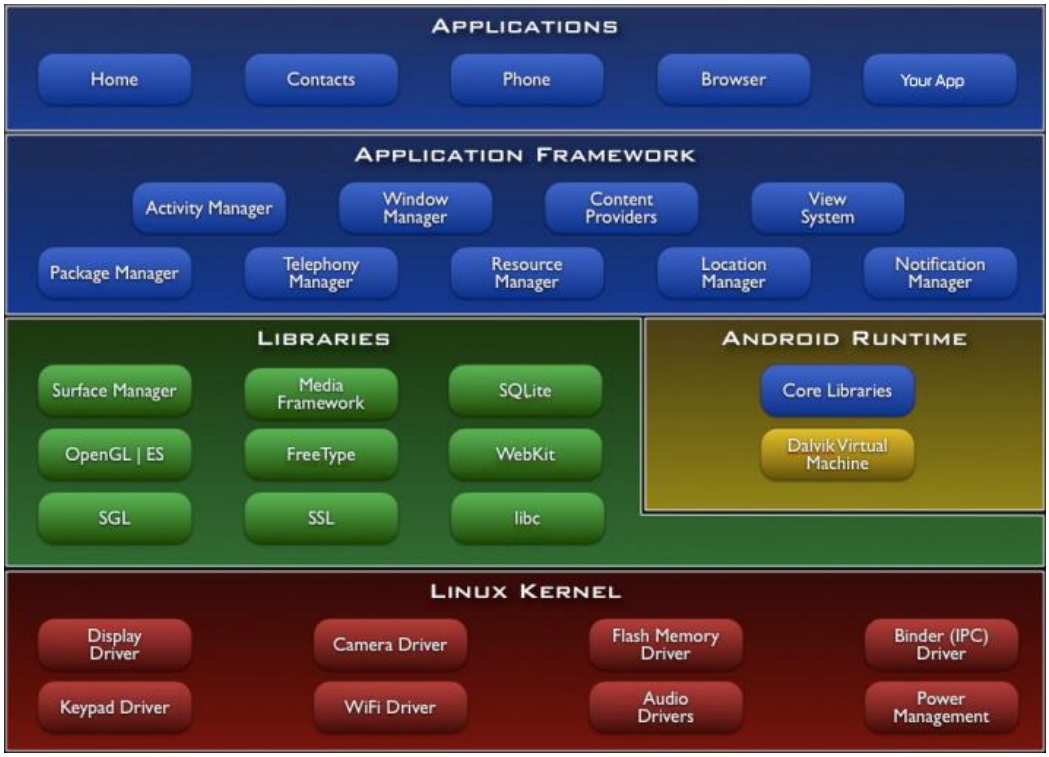

Ilustración 1 - Arquitectura de Android

Android incluye un gran conjunto de librerías a disposición de los desarrolladores. Muchas de ellas están directamente integradas en el marco de trabajo del sistema operativo de Google; algunos ejemplos son: las bibliotecas de 3D, las bibliotecas de medios, de gráficos, la "System C Library" o las bibliotecas de SQLite. No obstante, también existen bibliotecas de desarrollo externas que pueden utilizarse en Android, como Unity [2] , para desarrollo de videojuegos, o bibliotecas que pueden integrarse en el marco de trabajo, como OpenCV, las cuales conciernen al presente proyecto.

Android es, además, el sistema operativo con mayor cuota de mercado a nivel mundial. Esto, añadido al fácil (y gratuito) acceso al desarrollo, supone que sea la alternativa más interesante a la hora de elaborar una aplicación para smartphones.

Top Four Operating Systems, Shipments, and Market Share, Q3 2013 (Units in Millions)

Operating System	3Q13 Shipment Volumes	3Q13 Market Share	3Q12 Shipment Volumes	3Q12 Market Share	Year-Over-Year Change
Android	211.6	81.0%	139.9	74.9%	51.3%
iOS	33.8	12.9%	26.9	14.4%	25.6%
Windows Phone	9.5	3.6%	3.7	2.0%	156.0%
BlackBerry	4.5	1.7%	7.7	4.1%	-41.6%
Others	1.7	0.6%	8.4	4.5%	-80.1%
Total	**261.1**	**100.0%**	**186.7**	**100.0%**	**39.9%**

Ilustración 2 - 81% de cuota de mercado final 2013 (Fuente: IDC Worlwide Mobile PhoneTracker)

2.2. OpenCV

OPENCV – Open Source Computer Vision [3]– es un conjunto de librerías diseñadas para desarrollar programas que utilicen la visión por computador. Programadas en C++, su principal objetivo es el de establecer una infraestructura común para las aplicaciones basadas en procesamiento de imagen por computador, posibilitando, además, la utilización de estos avances en productos comerciales.

Están principalmente enfocadas al desarrollo de aplicaciones en tiempo real, aprovechando el procesamiento multihilo.

Contienen más de 2500 algoritmos optimizados y desarrollados para numerosos objetivos, entre los que se encuentran: detección de objetos, detección de movimiento, calibración de cámaras, tracking (seguimiento), visión estérea, visión robótica, reconocimiento de caras, extracción de modelos 3D de objetos, reconocimiento de escenarios, comparación de imágenes, etc.

Las librerías OpenCV son multiplataforma, por lo que se pueden encontrar versiones tanto para Linux como para Windows, Mac OS o Android. Además, es posible utilizar numerosos lenguajes a la hora de desarrollar con las librerías de OpenCV: Java, Python, C, C++ o MATLAB.

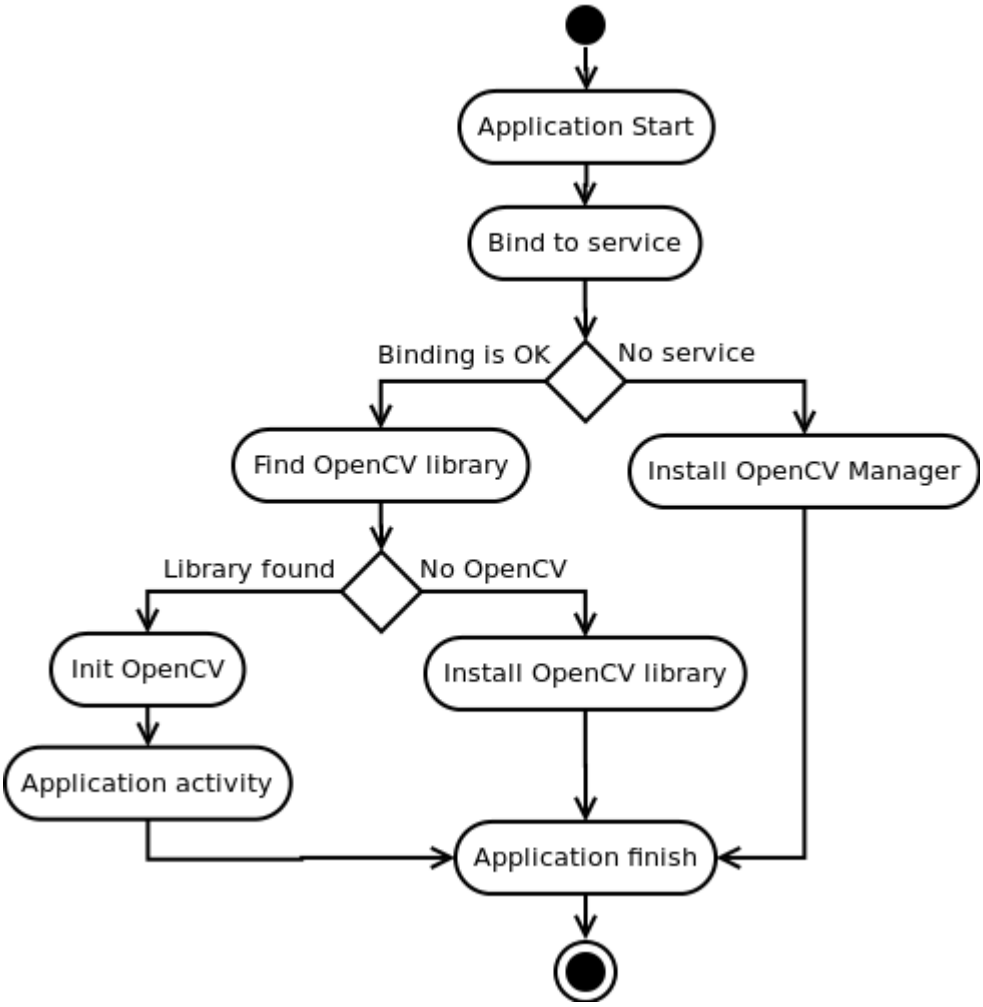

Figure 1 - Ejemplo de flujo de aplicación Android–OpenCV

Numerosas multinacionales como Google, Yahoo, Sony, Intel, IBM o Toyota utilizan estas librerías en sus productos. Se pueden mencionar numerosos ejemplos: en Israel, utilizadas en videocámaras de vigilancia que detectan el movimiento; en varios países europeos, utilizadas para detección de ahogamientos; ejecutando arte interactivo en museos de España y Nueva York o inspeccionando etiquetas de todo tipo de productos por todo el mundo.

2.3. ITS – Intelligent Transportation Systems

Los ITS, en español STI – Sistemas de Transporte Inteligente –, aplican las tecnologías de la información y de la comunicación al transporte. Son un conjunto de soluciones tecnológicas enfocadas a mejorar la operación y seguridad del transporte terrestre.

El objetivo de estas tecnologías es lograr un transporte más sostenible, esto es, más eficiente, más seguro y más limpio. Computadores, satélites, sensores y todo tipo de algoritmos juegan un papel cada vez más determinante en los sistemas de transporte.

Su principal innovación consiste en integrar las tecnologías existentes para crear nuevos servicios, nuevos instrumentos que podrán ser utilizados con diferentes propósitos y en diferentes condiciones.

Es posible encontrar ITS en todo tipo de transportes:

- Carretera
- Ferrocarril
- Aire
- Mar

Cabe destacar que este tipo de tecnología se implanta tanto en vehículos de uso personal o transporte de pasajeros como en vehículos dedicados al transporte de mercancías.

La comunicación utilizada por los vehículos de transporte inteligente puede ser:

- Interacción con el entorno: utilización de sensores, algoritmos, etc.
- Interacción con otros vehículos: comunicación entre coches.
- Interacción con infraestructuras: vehículo en autopista de peaje.

Actualmente, ya se pueden encontrar vehículos ITS en el mercado, como son algunos modelos de marcas como Volvo o Mercedes. Dichos ejemplos se comentarán más ampliamente en el apartado "ITS en el mercado".

¿Qué beneficios pueden aportar estas tecnologías, tanto actualmente, como en los años venideros? He aquí algunos ejemplos:

- Reducción de tiempo de espera en semáforos en función de los peatones o vehículos que haya para cruzar.
- Reducción del tiempo de espera en vías interurbanas cuando un accidente ocurre.
- Cobro automático de peajes, de manera que se reduzcan las retenciones y la contaminación.
- Sistemas de navegación que recogen todo tipo de información en tiempo real, permitiendo al conductor decidir de manera más eficiente cómo, cuándo y hacia dónde viajar.
- Sistemas avanzados de tránsito que permiten a las empresas operar de forma más eficiente, proveyendo a sus clientes de información en tiempo real que haga sus viajes más fáciles y atractivos.

En definitiva, para los usuarios se logran viajes más placenteros, seguros y rápidos. Para las empresas, abaratamiento del coste de los transportes, mayor seguridad y eficiencia.

Ilustración 3 - ITS en el mundo real

Un gran número de países ya tienen sus propias instituciones dedicadas a la expansión y el desarrollo de los ITS. En la web de la Comisión Europea se puede encontrar una sección dedicada a los transportes inteligentes [4] y países como EEUU [5], Francia [6], o Reino Unido [7] tienen también instituciones estatales. En EEUU existen incluso diferentes instituciones de ITS por estado, como por ejemplo las de Nueva York [8] u Oregon [9].

En España existe el "Foro de Nuevas Tecnologías en el Transporte, ITS España" [10] dedicado a aunar los esfuerzos del sector público, privado y académico en torno a los transportes inteligentes. En su web aparecen diferentes actividades, conferencias, documentación, etc.

2.4. SLAM – Simultaneous Localization and Mapping

En este sub-apartado se realiza una breve introducción sobre la técnica de SLAM, la cual tiene relación con la aplicación de "Detección de movimiento" en lo que a reconocimiento del entorno se refiere.

SLAM es una técnica utilizada por dispositivos digitales cuya labor consiste en construir mapas de un entorno desconocido, permitiendo al robot situarse y desplazarse por el mismo. Realizando diferentes aprendizajes y utilizando la información obtenida a través de sus cámaras y/o sensores, la máquina es capaz de reconocer el entorno, detectar cambios en el mismo, desplazarse por él, evitar obstáculos, etc.

Hay numerosos pasos en las diferentes formas de implementar SLAM y en dichos pasos se pueden usar, a su vez, diferentes algoritmos. Sin embargo, se han de definir cinco etapas clave en la implementación de SLAM:

- Reconocimiento del entorno.
- Procesamiento de datos.
- Estimación del estado.
- Actualización del estado.
- Actualización del entorno.

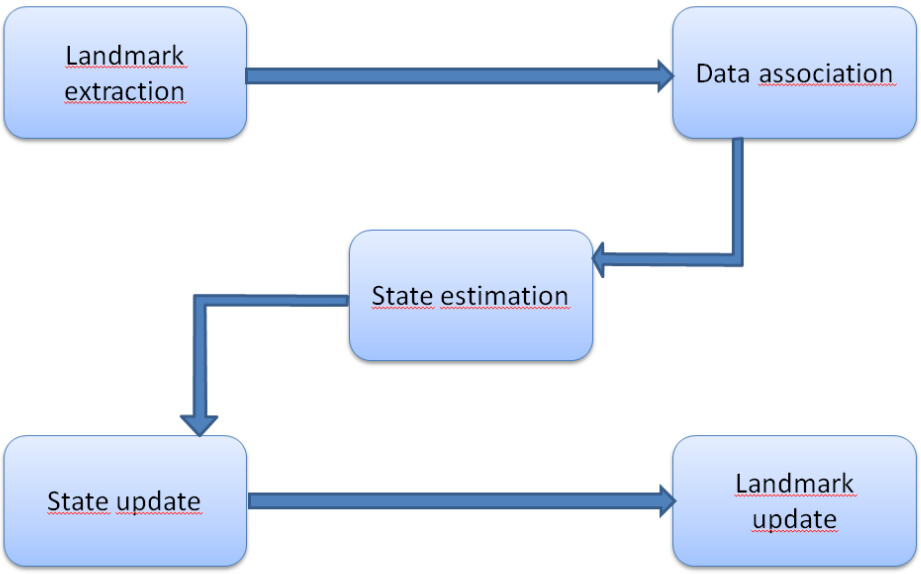

Ilustración 4–Diagrama de funcionamiento de SLAM

2.5. OpenCV en la Play Store de Google

Si se realiza una búsqueda en la Play Store con la palabra "OpenCV", se comprobará que aparecen unos doscientos resultados. Sin embargo, muchos de ellos no son más que comercializaciones de los ejemplos de OpenCV, como el detector de caras, el puzzle con la imagen recogida en vídeo o los histogramas de color en tiempo real.

No hay ninguna aplicación que trate la detección de peatones.

Pocas aplicaciones tienen un objetivo estratégico, si acaso ejemplificativo. Entre las numerosas aplicaciones halladas, se ha decidido destacar dos con puntos en común con las desarrolladas en este proyecto.

2.5.1. OpenCV Movement Detect

Aplicación publicada en Febrero de 2014, es un ejemplo de detección de movimiento. Cómo se puede apreciar en la ilustración 5, la ejemplifican como cámara de seguridad. Se encuentra en un punto fijo y marca en pantalla los puntos en los que se produce movimiento.

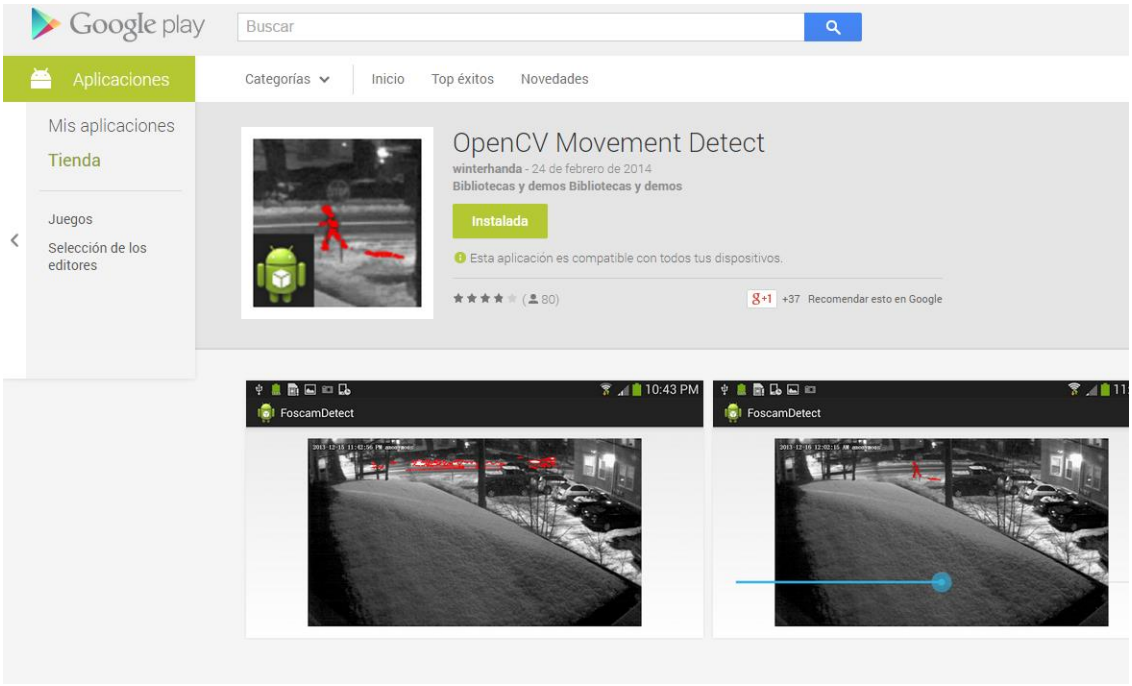

Ilustración 5 - Aplicación OpenCVMovementDetect

Esta aplicación se diferencia en que no detecta el movimiento de la misma manera que la desarrollada en este proyecto, ya que cuenta con una barra de progreso, "seekbar" en

Android, que permite al usuario configurar el detector para obtener un mayor o menor rango de detección.

En la aplicación de este proyecto se establece un ratio de detección constante, para evitar posibles fallos de detección debido a una configuración inadecuada de dicha característica de contraste.

2.5.2. ButtuCarRun

Este es un buen ejemplo de aplicación ITS, con una idea similar a las de este proyecto. Es una aplicación japonesa cuyo objetivo principal es alertar al conductor de posibles colisiones. A través de diferentes algoritmos, calcula la distancia del coche o coches más cercanos al usuario, pitando en el caso de que la distancia sea inferior a cinco metros.

Ilustración 6 - Aplicación ButtuCarRun

En ambos casos, se ha podido comprobar que ninguna de las aplicaciones produce resultados superiores a los ofrecidos por la aplicación desarrollada en este proyecto, teniendo en cuenta las relativas similitudes con estas aplicaciones.

Un punto clave a destacar es la fecha de publicación de ambas aplicaciones, Febrero de 2014. Es muy importante en relación al inexplorado campo de OpenCV en Android que se recalca en esta memoria.

2.6. ITS relacionados en el mercado

En este sub-apartado se analizarán algunos de los ejemplos más destacados de vehículos inteligentes disponibles en el mercado y que cuenten con detección de peatones.

La empresa sueca Volvo es una de las pioneras en el desarrollo de ITS para el mercado comercial de vehículos. Actualmente, usted puede comprar vehículos que incorporan detección de peatones e incluso ciclistas y frenan automáticamente cuando la distancia se reduce hasta un límite considerado peligroso.

En este vídeo de 2010, Volvo S60 Pedestrian Detection Test: [11], se muestran una serie de pruebas del sistema de detección de peatones de Volvo. Como se puede apreciar, no es infalible.

Ilustración 7 - Fotograma del momento del atropello

Posteriormente se presentó Volvo Car 2 Car communication [12], vídeo que muestra las ventajas ITS de Volvo respecto a la comunicación entre vehículos:

- Detección de vehículo accidentado.
- Detección de vehículo de emergencias.
- Detección cuando otro coche se salta un semáforo en rojo.

Ilustración 8 – Detección de posible colisión, frenado automático

Ya en 2013, Volvo presentó un nuevo sistema de detección de ciclistas y peatones [13].

También se puede encontrar otro ejemplo de ITS en Mercedes Pre Safe [14], que cuenta con ajuste automático de los sistemas de seguridad del coche (cinturones, airbags, asientos, etc.) cuando el sistema detecta un peligro inminente y con frenado automático al detectar un peatón.

Toyota es otra de las compañías que trabaja más activamente en sistemas de detección de peatones. En pruebas de este mismo año [15], se demostró que a una velocidad superior a 25 mph (el equivalente a unos 40 km/h) los resultados no son concluyentes.

Toyota's pedestrian-detection system makes an impact at the Consumer Reports track

But sometimes hits Steve

Published: June 05, 2014 01:00 PM

Ilustración 9 - Momento del atropello a 40 Km/h

Todos estos ejemplos demuestran que, si bien el mercado de los ITS es un mercado en alza, todavía queda mucho camino por recorrer. También afianza la tesis sobre la dificultad de realizar una buena detección en un dispositivo smartphone.

2.7. Proyecto relacionados del LSI

El Laboratorio de Sistemas Inteligentes de la UC3M trabaja muy activamente en nuevos sistemas relacionados con el campo de la automoción y, en concreto, con la detección de peatones. El LSI fue premiado recientemente, junto con miembros de la UPM, con el "Premio a la investigación en el campo de la automoción", de la fundación Eduardo Barreiros. [16]

Uno de los sistemas más destacados del LSI es el coche inteligente que es capaz de detectar peatones de noche o en condiciones de baja visibilidad [17]. El proceso de detección se basa en la captación del calor corporal de los peatones a través de un conjunto de cámaras infrarrojas.

Ilustración 10 - Fotograma del detector desarrollado por el LSI

Como se explica en la propia web del LSI, en el enlace dedicado a este proyecto [18], los diferentes clasificadores utilizados cuentan con miles de imágenes.

Además de este proyecto, se pueden encontrar numerosos artículos relacionados con la detección de peatones en la sección de publicaciones [19].

3. SOFTWARE

3.1. Elección del entorno de desarrollo

Para realizar el desarrollo de la aplicación en Android se tuvieron en cuenta dos posibles plataformas de desarrollo:

- Android Studio. Un novedoso entorno de desarrollo integrado, lanzado a mediados de 2013.
- SDK de Android. El kit de desarrollo de software de Android para Eclipse, el más común a la hora de desarrollar aplicaciones para este sistema operativo.

En las primeras conversaciones con Fernando, tutor del proyecto, y Juan, director del mismo, se valoró la idea de desarrollar la aplicación en la plataforma Android Studio. Dicho planteamiento surgió en relación a la utilización de software lo más novedoso posible, ya que el desarrollo de las OpenCV en Android es muy reciente y se pensó que podría ser muy útil aprender a desarrollar en la nueva plataforma para programar en Android.

Como toda persona relacionada con el mundo de la tecnología sabe, el software novedoso suele conllevar problemas a la hora de realizar labores que ya estaban estandarizadas en programas similares. También es común la aparición de bugs y errores que se van puliendo en las versiones posteriores. Cabe destacar, además, que Android Studio se encontraba en versión "beta" en la fecha de inicio del proyecto (Febrero 2014).

La otra posibilidad barajada fue la de desarrollar la aplicación con el SDK de Android para Eclipse, el más utilizado y del que más información se disponía, con motivo de realizar búsquedas exhaustivas y técnicas dirigidas a entender mejor OpenCV en Android.

Finalmente, la decisión fue la de escoger el Android SDK, por los motivos expuestos en este apartado y por la existencia de algunos problemas con Android Studio, documentados con más precisión en el apartado "Problemas".

3.2. Versiones de Android utilizadas

La versión objetivo ("target-version" en Android) es la API 19, Android 4.4 - KitKat [20] , distribuida el 31 de octubre de 2013.

Sin embargo, no es tan importante la versión objetivo como la versión mínima en la que la aplicación funciona correctamente, puesto que en versiones anteriores de Android no se asegura la adecuada ejecución de todas las características de la aplicación.

OpenCV sólo funciona a partir de la API 8, Android 2.2 - Froyo, lanzada en Mayo de 2010. Durante los primeros meses se estableció la API 10, Android 2.3 –Gingerbread como la mínima adecuada pero, tras la implementación de los botones de vibración, era imprescindible el cambio a la API 13, Android 3.2 – Honeycomb.

3.3. Clasificador Haar-Cascade

Los clasificadores Haar-Cascade forman parte imprescindible en las aplicaciones de detección de objetos tratadas en este documento.

Este tipo de archivos, generados en formato XML, fueron propuestos por Paul Viola y Michael Jones en 2001.Su denominación completa es "cascade of boosted classifiers working with haar-like features".

El desarrollo del clasificador consiste en un entrenamiento con varios cientos de imágenes positivas (contienen el objeto a detectar) y negativas (imágenes aleatorias que no deben contener el objeto a detectar). Todas estas imágenes positivas son re-escaladas a un tamaño tipo y, en ellas, se determina el objeto a detectar. Las imágenes negativas simplemente son re-escaladas.

La palabra "cascade" (cascada) se refiere a que el resultado del clasificador proviene de numerosos clasificadores simples, conocidos como "stages" (etapas). Dichos "stages" son aplicados de manera sucesiva sobre una región de interés hasta que uno de ellos es rechazado o todos son aceptados.

El método "boosting" se utiliza para la construcción de clasificadores en cascada. Este método consiste en que los clasificadores son complejos independientemente de cada etapa y que están construidos usando una "boostingtechnique" conocida como "weightedvoting" (votación ponderada).

Los clasificadores básicos utilizan un árbol de decisión de, al menos, dos hojas. Las características Haar de entrada a los clasificadores básicos se calculan como se describe en la imagen:

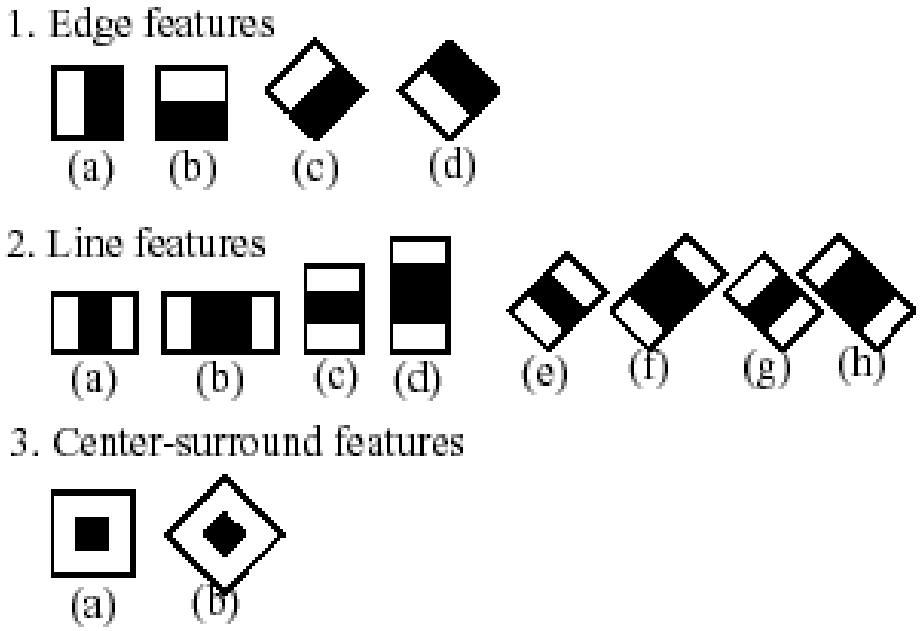

Ilustración 11 - Filtros con Base Haar

La principal característica utilizada en un clasificador particular viene especificada por la forma del objeto a detectar, la posición dentro del área de interés y el tamaño. Los filtros con base Haar realizan una codificación calculando la diferencia de intensidades mediante la captura de contraste entre diferentes regiones de la imagen. De esta forma, se generan contornos, puntos y líneas.

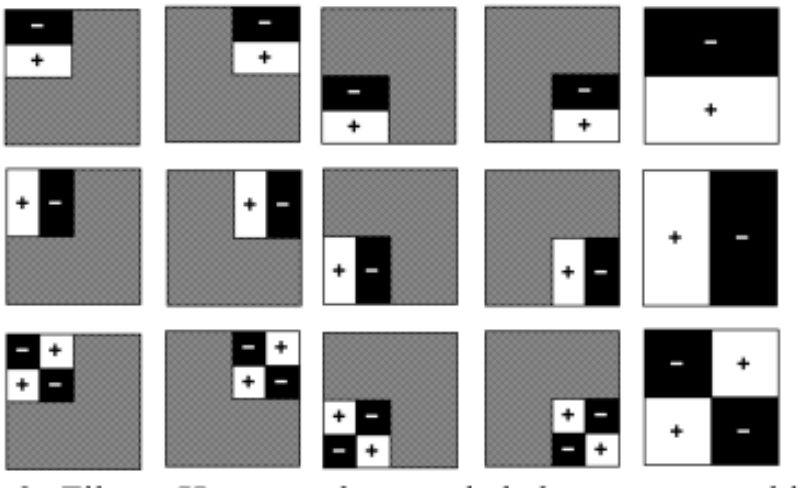

Ilustración 12 - Filtros Haar: rotación, traslación y re-escalado

Por tanto, el flujo de entrenamiento de un clasificador Haar se puede definir mediante cinco etapas:

- Introducción de una imagen de entrada.
- Re-escalado y transformación de la imagen mediante una serie de operaciones básicas. La imagen resultante se denomina imagen integral.
- Extracción de las características de la imagen: procesamiento de las características de la imagen, realizando un reconocimiento de patrones. En este paso se utilizan los filtros Haar, que generarán las características de contornos, puntos y líneas.
- Clasificación de la imagen: en esta etapa se asigna un conjunto de características a la imagen de acuerdo al modelo inducido durante el entrenamiento.
- Finalización: se obtiene la imagen con el objeto detectado.

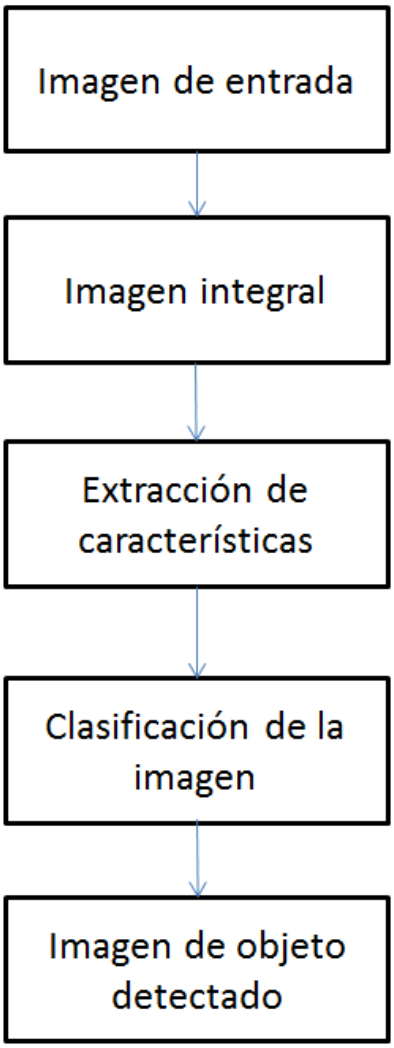

Ilustración 13 - Flujo de entrenamiento de un clasificador Haar

Para generar un clasificador Haar-Cascade se necesita una gran cantidad de imágenes positivas y negativas. Es decir, son necesarias al menos unas 5000 imágenes positivas del objeto que se quiere detectar por un lado y, por otro lado, unas 2000 imágenes negativas (sin el objeto a detectar).

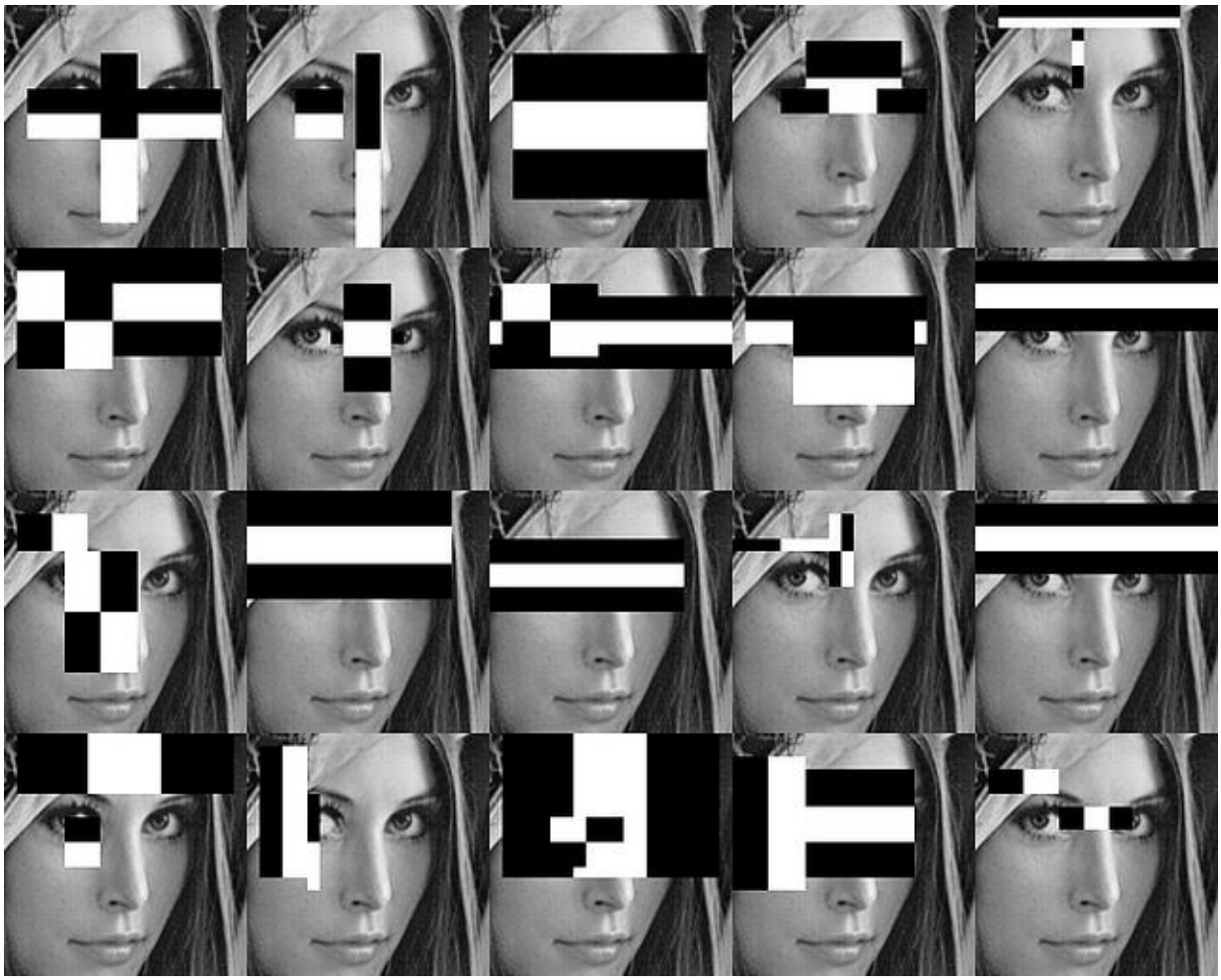

Ilustración 14 - Ejemplo de clasificación Haar

Una vez generado el clasificador con los valores tanto de imágenes positivas como negativas, se cargará en un fichero que la aplicación utilizará para detectar objetos mientras se mantenga la ejecución.

3.4. La aplicación base

Ilustración 15 - Interfaz inicial

La aplicación base, contenedora del resto de "subaplicaciones", recibe el nombre de "LSIopenCV". Esta base contiene cinco botones iniciales.

- Primer botón – Logo del LSI: enlaza con la página web del LSI – UC3M. En dicha web se puede consultar todo lo relacionado con el departamento: miembros, publicaciones, proyectos actuales, etc.

Ilustración 16 - Botón LSI

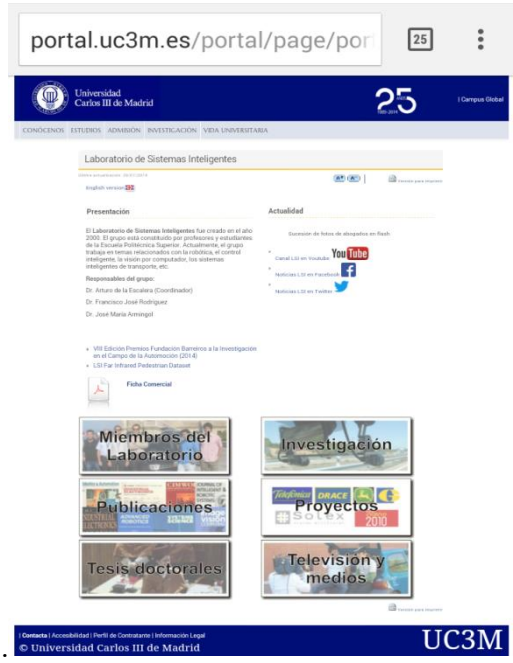

Ilustración 17 - Web del LSI

- Segundo botón – Aplicaciones LSI: este botón conduce a una nueva interfaz en la cual el usuario podrá desplegar un menú con todas las aplicaciones contenidas en la aplicación base, eligiendo la que considere oportuna.

Ilustración 18 - Aplicaciones del LSI

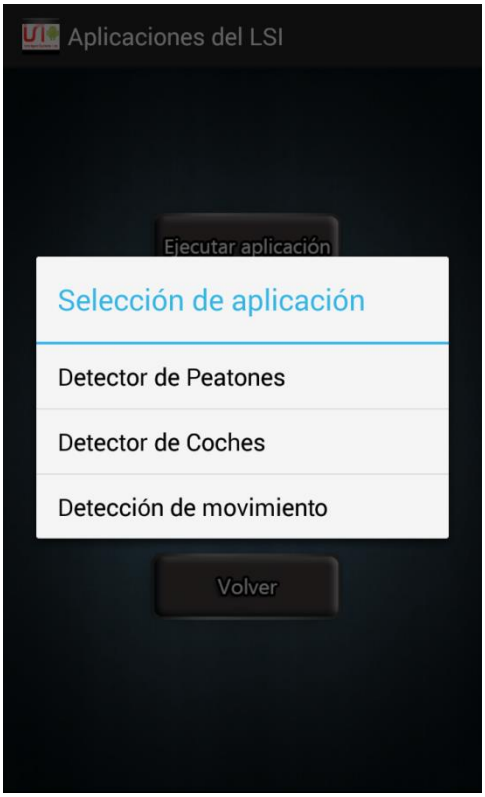

Ilustración 19 - Menú de selección

- Tercer botón – Ejemplos: al pulsar a este botón, se accede a una activity que contiene tres ejemplos de uso de las OpenCV, los cuales se pueden obtener de manera sencilla y gratuita en la web de OpenCV [21].

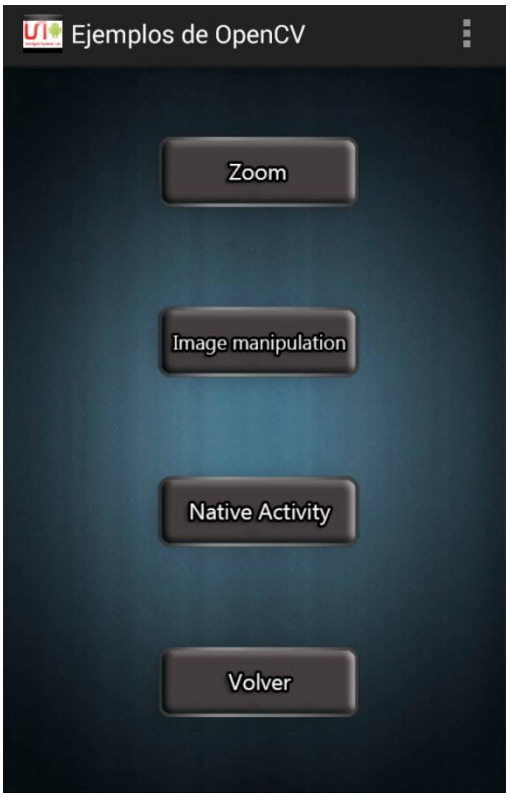

Ilustración 20 - Ejemplos OpenCV

- Cuarto botón - ¿Quiénes somos?: este botón lleva a una sencilla interfaz en la que se puede leer el nombre del autor de la aplicación, del tutor y el director de proyecto. Además, se facilitan sus correos electrónicos.

Ilustración 21 - ¿Quiénes somos?

- Quinto botón – Logo de la UC3M: este botón enlaza directamente con la página web de la universidad: http://www.uc3m.es .

Ilustración 22 - Logo UC3M

En los siguientes apartados se explicarán más detalladamente las diferentes aplicaciones que contiene el proyecto "LSIopenCV".

3.5. Detección de peatones

La aplicación de detección de peatones consiste en, como su propio nombre indica, notificar al usuario la presencia de un peatón cuando éste se encuentra en el encuadre de la cámara del Smartphone.

Esta aplicación basa su funcionamiento en el uso de un clasificador Haar-Cascade para la detección de peatones. Dicho clasificador se carga en un fichero propio de la activity.

Una vez se activa la cámara, el programa empieza a procesar imagen por imagen.

- Procesamiento de imagen.
- Re-escalado.
- Algoritmo de detección.
- Encuadre de los objetos detectados.

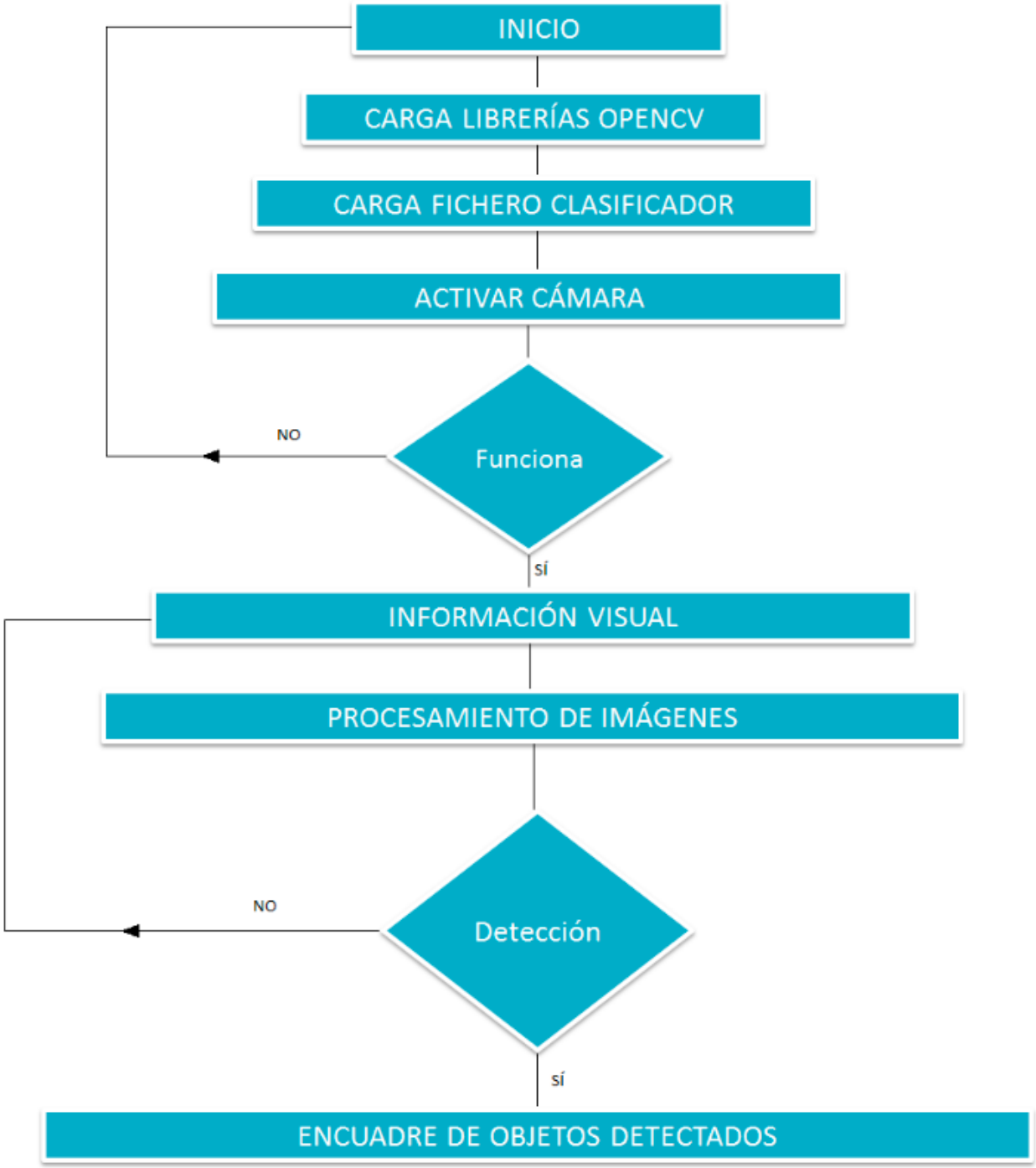

Ilustración 23 - Flujo de detección

Es extremadamente importante configurar debidamente los algoritmos de detección. La limitación tecnológica, tratada con mayor profundidad en el apartado "Limitaciones",

provoca que, si se aumenta la tasa de imágenes por segundo, disminuya la precisión del detector (aumento de falsos positivos). Esta situación hace necesario alcanzar un equilibrio.

Dentro de la aplicación el usuario encontrará dos botones:

- Activar/desactivar vibración: esta función alerta cuando un peatón es detectado.
- Porcentaje de re-escalado: establece, de manera porcentual, el tamaño mínimo del peatón a detectar. Si se pretende detectar peatones más pequeños, se debe procesar la imagen durante más tiempo. Por tanto, a menor porcentaje, menor framerate pero mayor precisión.

  Se puede variar entre: 50%, 40%, 30% y 20%, siendo éste último el más preciso.

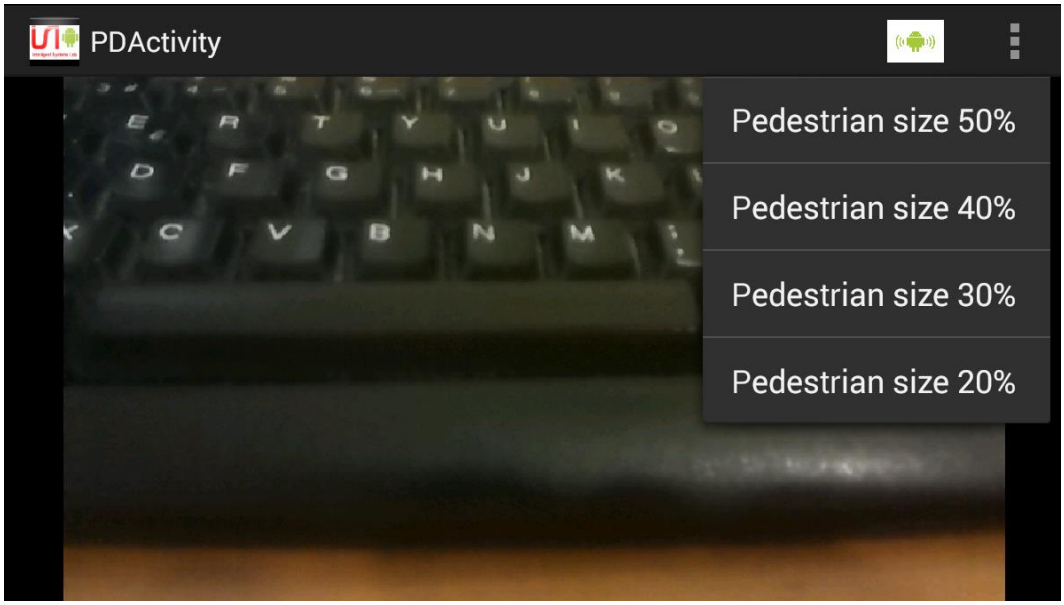

Ilustración 24 - Botones de re-escalado y vibración

3.5.1. Framerate

Para la realización de esta tabla, se han utilizado los valores de re-escalado de más precisión. En este caso, el 20%.

Detección de peatones		
Dispositivo	**Resolución**	**Framerate**
Motorola Moto G	864*480	8 FPS
HTC Desire	480*320	5,50 FPS
Nexus 5	1280*768	8 FPS
Tablet Asus Nexus 7	1280*768	8 FPS

Tabla 2 - Tasa de FPS por dispositivo

3.5.2. Pruebas

Tras muchas horas de pruebas, los resultados obtenidos han sido verdaderamente satisfactorios, si bien, se han de tener en cuenta los siguientes aspectos:

- Clasificador no adaptado al dispositivo. Esto limita la capacidad de detección ya que, en un caso ideal, el entrenamiento debería haberse realizado con el propio móvil.
- Limitaciones en el framerate. En el mejor de los casos, la tasa de imágenes por segundo ronda los 10-12 FPS.
- Condiciones muy concretas. Las condiciones que deben darse en la imagen para que el objeto sea correctamente detectado:
    - Que el peatón esté lo suficientemente cerca, pero aparezca completo en la imagen.
    - Fuerte contraste con el fondo.
    - Posición de peatón.
    - Espacios bien iluminados.

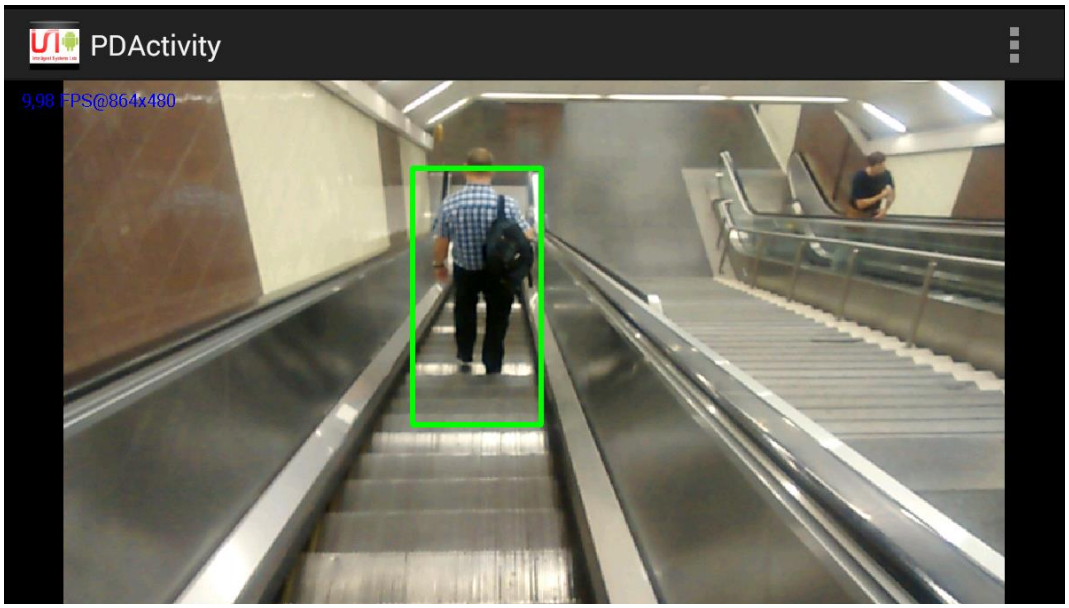

Ilustración 25 - Entorno bien iluminado

Como se ve en ilustración 25, la postura del peatón unida a una buena iluminación, facilita la detección.

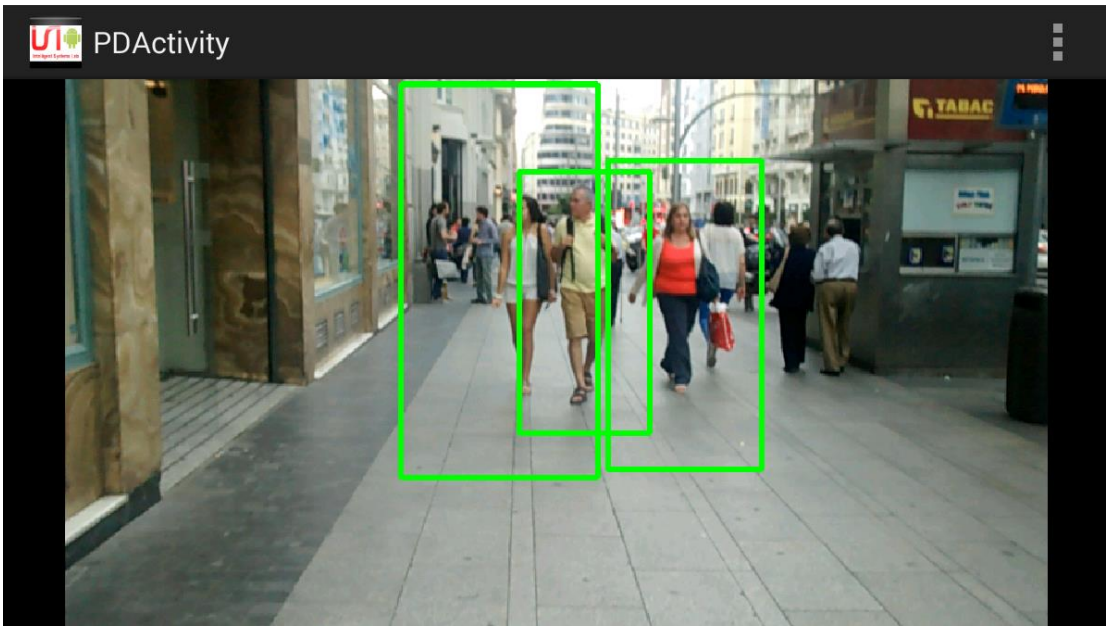

Ilustración 26 - Ejemplo de multidetección

Ilustración 27 - Peatón con ropa de contraste

El contraste en la ilustración 27 es clave para que se realice una detección positiva.

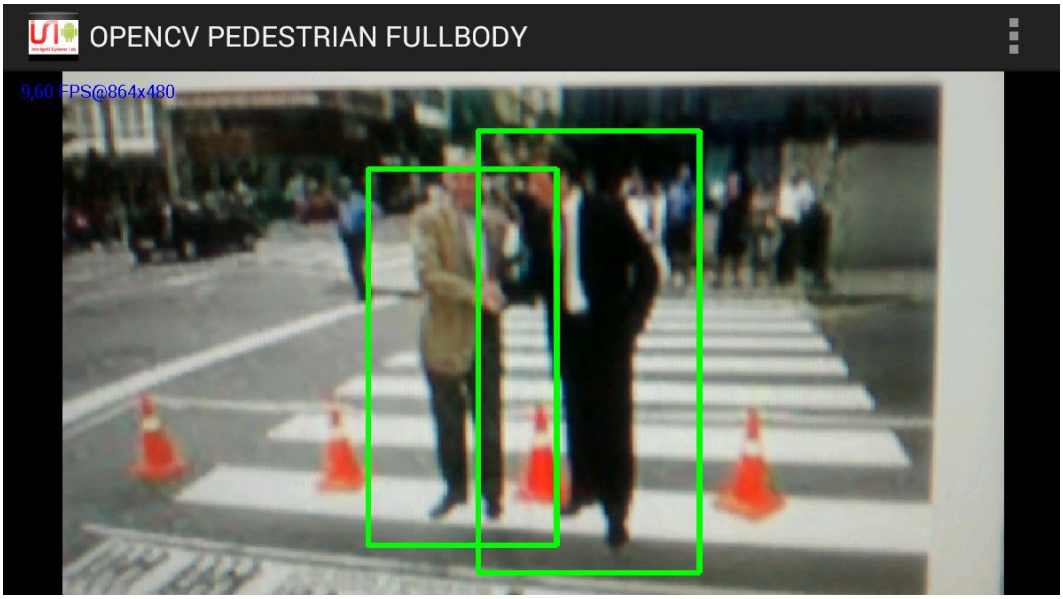

Ilustración 28 - Multidetección a baja resolución

En la ilustración 28 se observa que, a pesar de ser una imagen en baja resolución, si la posición de los peatones y el contraste con el entorno son adecuados, la detección será positiva.

3.6. Detección de vehículos

La aplicación de detección de vehículos avisa al usuario mediante vibración cuando realiza la detección de un vehículo.

Es importante señalar que el clasificador Haar-Cascade utilizado está entrenado para detectar coches principalmente por la parte trasera de los mismos. Dicho clasificador, al igual que el de detección de peatones, fue facilitado por el tutor, Fernando García.

Esta aplicación sigue un funcionamiento muy similar al de la aplicación de detección de peatones.

- Carga de librerías OpenCV.
- Carga de fichero clasificador.
- Activación de cámara.
- Procesamiento de imagen.
- Detección de objetos, en este caso vehículos.

La principal diferencia radica, aparte de en el clasificador, en la configuración de las funciones de detección.

En esta aplicación se disponen dos botones:

- Activar/desactivar vibración: esta función alerta al usuario en el momento en el que se produce la detección de un vehículo.
- Porcentaje de re-escalado: en esta aplicación, al igual que en la de detección de peatones, a menor porcentaje, mayor precisión. Sin embargo, se ha decidido establecer únicamente tres valores de re-escalado: 40%,30% y 20%. Esto se debe a que los vehículos son objetos muy superiores en tamaño a los peatones, por lo que es extremadamente complicado detectarlos en el mayor porcentaje de re-escalado, 50%.

3.6.1. Framerate

Para calcular los valores de esta tabla, se ha utilizado como valor de re-escalado el de mayor precisión: 20%.

Detección de vehículos		
Dispositivo	**Resolución**	**Framerate**
Motorola Moto G	864*480	4'5-5'5 FPS
HTC Desire	480*320	2 FPS
Nexus 5	1280*768	4'5-5'5 FPS
Tablet Asus Nexus 7	1280*768	4'5-5'5 FPS

Tabla 3 - Tasa de FPS de detección de vehículos

3.6.2. Pruebas

En el caso de esta aplicación, es importante señalar:

- Problemas con el clasificador: este clasificador no es tan eficaz como el de detección de peatones. Tras muchas horas de prueba, se ha conseguido establecer con unos resultados satisfactorios.
- Framerate: si el teléfono está en un punto fijo, el framerate es mayor, permitiendo una detección mejor que en movimiento.

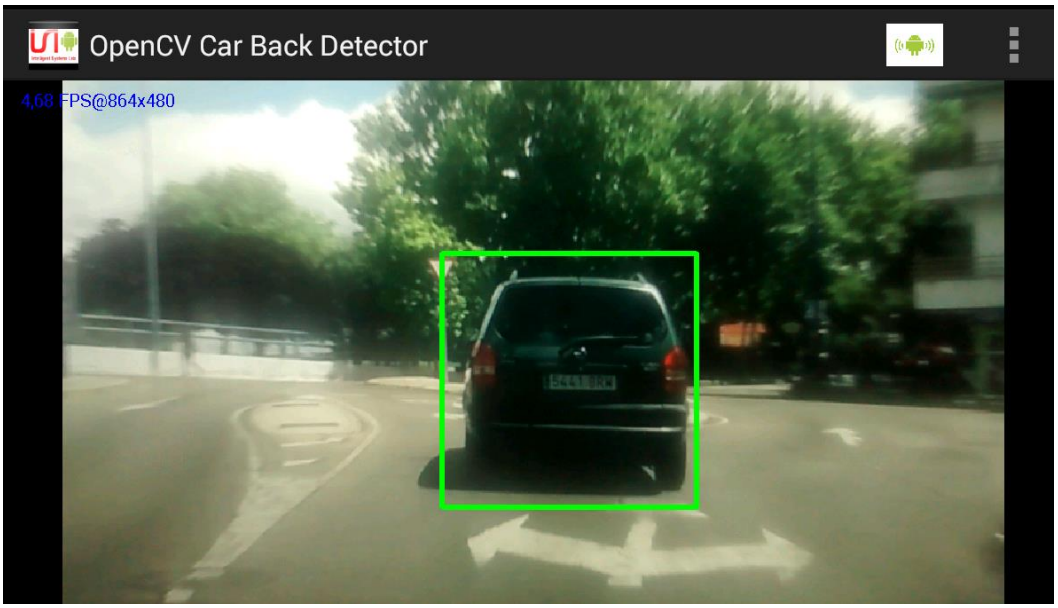

Ilustración 29 - Detección realizada desde nuestro vehículo

Como se puede apreciar en la ilustración 29, manteniendo una distancia adecuada se realiza una buena detección.

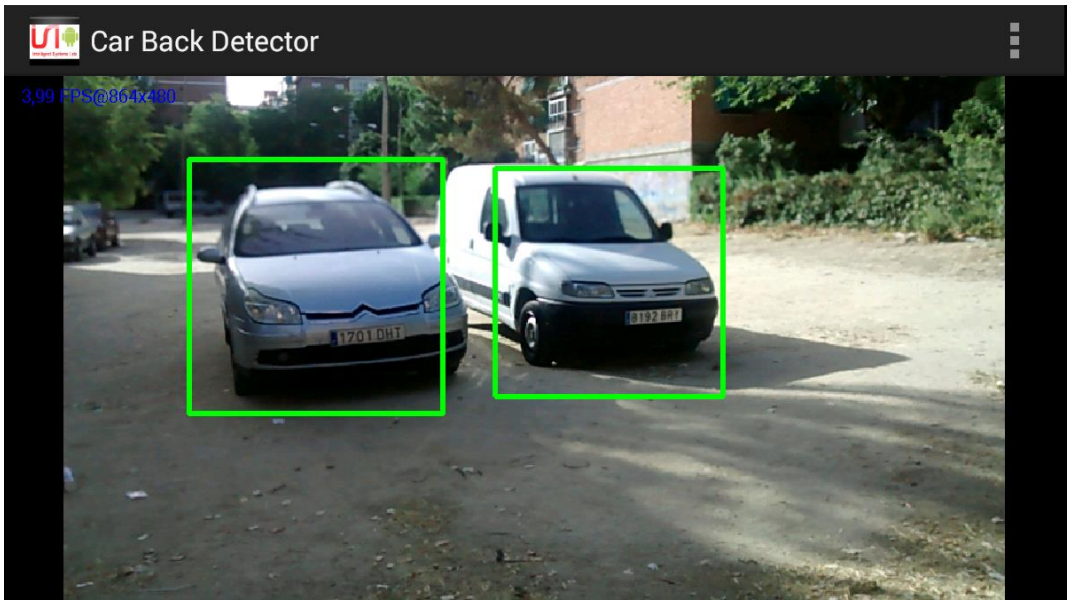

Ilustración 30 - Detección múltiple

La ilustración 30 es un ejemplo de cómo una buena iluminación y contraste facilitan la detección.

3.7. Detección de movimiento

La aplicación de detección de movimiento advierte al usuario mediante vibración cuando se produce un movimiento en la imagen.

OpenCV provee algoritmos que comparan las imágenes consecutivas que reciben a través de la cámara, sustrayendo los objetos en movimiento. Estos algoritmos se conocen como "Algoritmos sustractores".

OpenCV soporta tres tipos de algoritmo sustractor:

- MOG
- MOG2
- GMG

Ilustración 31 - Ejemplo de aplicación de algoritmos sustractores

En esta aplicación se utiliza el algoritmo MOG2. Como se observa en la ilustración 31, este sustractor también extrae las "sombras" de los objetos, en la imagen los puntos grises.

El flujo de funcionamiento de la aplicación es el siguiente:

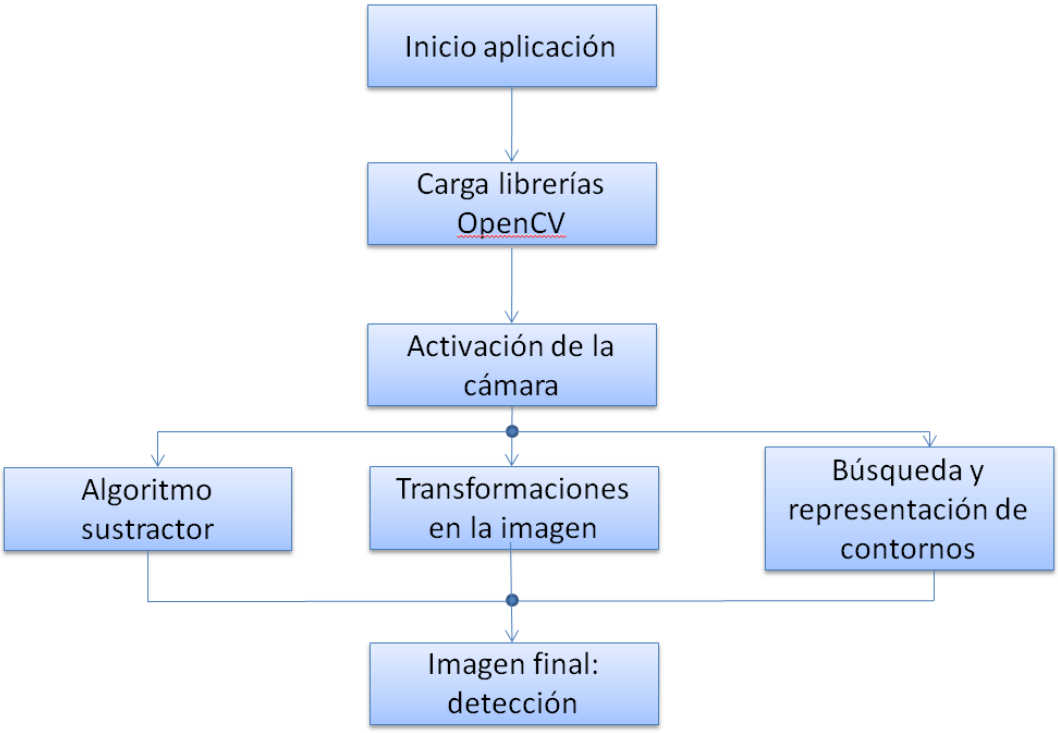

Ilustración 32 - Flujo de detección de movimiento

3.7.1. Framerate

En la tabla 4 se analizan los FPS resultantes en esta aplicación para los cuatro dispositivos de prueba.

Detección de movimiento		
Dispositivo	**Resolución**	**Framerate**
Motorola Moto G	864*480	4-5 FPS
HTC Desire	480*320	2'5 FPS
Nexus 5	1280*768	4-5 FPS
Tablet Asus Nexus 7	1280*768	4-5 FPS

Tabla 4 - Tasa de FPS detección de movimiento

Como se puede observar en la tabla 2, 3 y 4, las diferencias sustanciales sólo se dan con el dispositivo más antiguo, muy inferior a los otros tres.

3.7.2. Pruebas

En este apartado se muestran algunas pruebas realizadas con la aplicación de detección de movimiento.

En la ilustración 33 se observa una configuración diferente a la de la ilustración 34. En la primera imagen el granulado de detección de movimiento es más grueso, mientras que en la segunda, posterior en el tiempo, mejora el framerate.

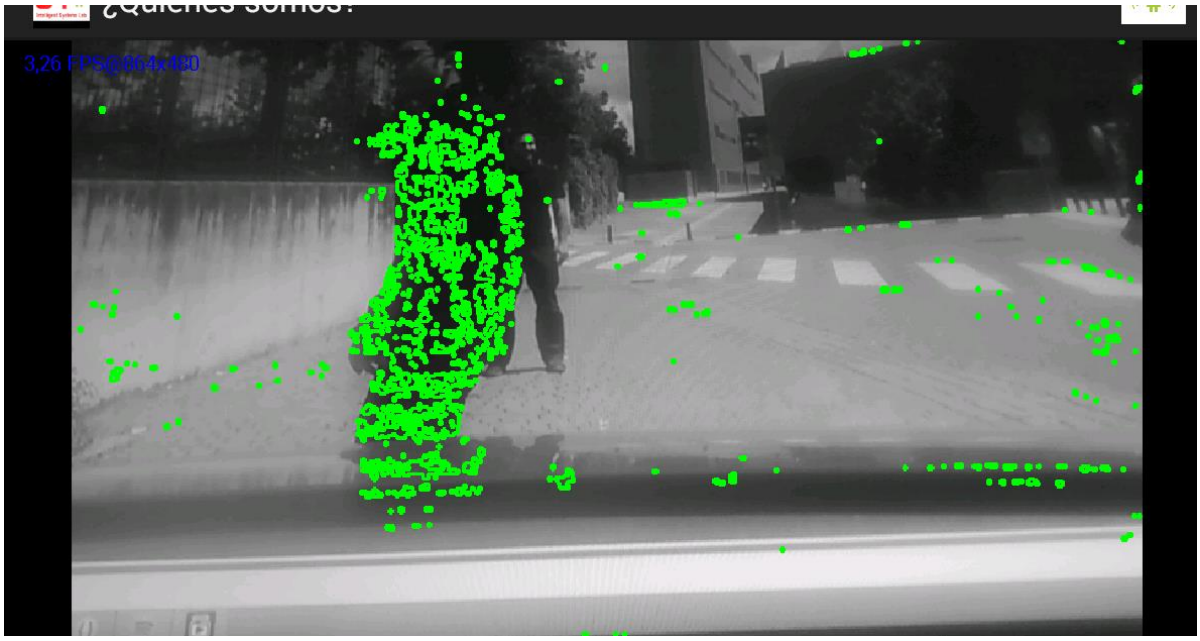

Ilustración 33 - Detección de movimiento (andando)

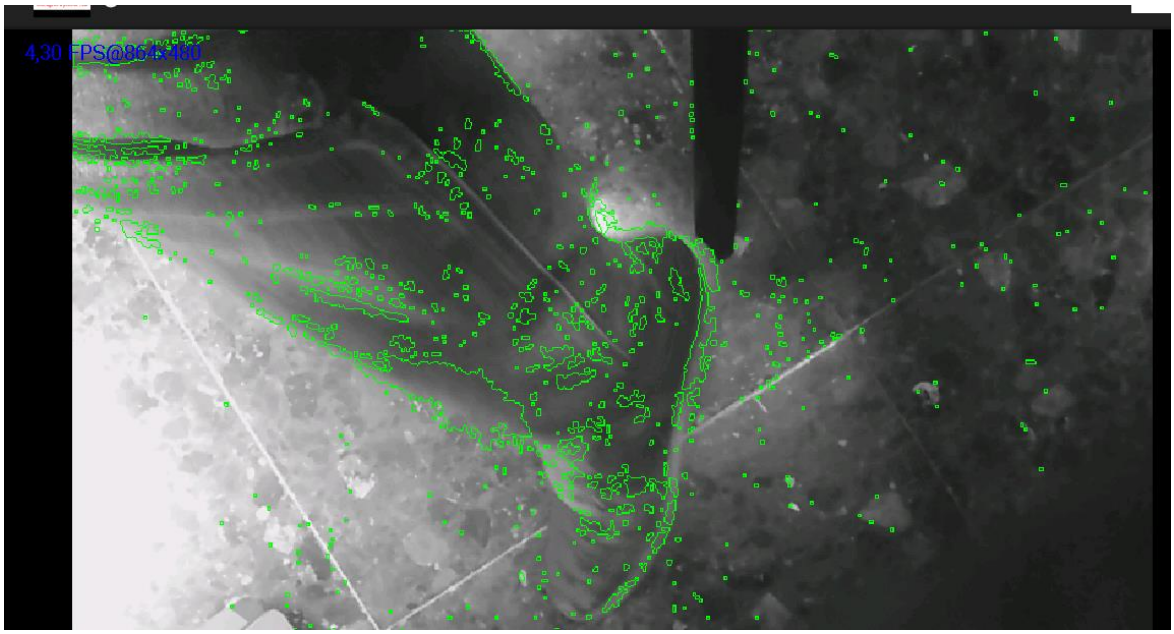

Ilustración 34 - Detección de movimiento

4. LIMITACIONES

Es extremadamente importante entender las limitaciones que supone, para un proyecto enfocado a la visión por computador, la utilización de un dispositivo smartphone.

- Calidad de la cámara.
- Resolución.
- Condiciones de luz o climatológicas.
- Fluidez en la detección de imagen.
- Características del clasificador

4.1. Calidad de la cámara

Como ya se ha explicado detenidamente en esta memoria, la detección de peatones es un proceso en evolución, cuyos resultados mejoran año tras año, pero aún lejos de la perfección. Si a esto se añade una cámara de Smartphone, es posible comprender las limitaciones que ésta impondrá a la hora de enfocar, de reconocer patrones, etc.

4.2. Resolución

La resolución del dispositivo también es un dato a tener en cuenta. Por defecto, se muestra arriba a la izquierda la resolución, además de los FPS (imágenes por segundo).

Los peatones/vehículos deben encontrarse a una distancia ideal de la cámara para ser eficazmente reconocidos.

4.3. Condiciones de luz

La detección de formas se realiza con mayor facilidad en ambientes con buena iluminación. En concreto, la detección de peatones se facilita en días soleados, aunque también funciona, no con la misma eficacia, en días lluviosos.

Cuando es de noche, la detección es prácticamente nula. Esto se debe a que el algoritmo de detección reconoce mejor los patrones cuanto mayor es el contraste.

De cara a trabajos futuros, se podría tener en cuenta la posibilidad de mejorar la aplicación utilizando algún tipo de sensor o visión nocturna que permita la detección de formas en lugares con poca luz o en condiciones climatológicas adversas.

4.4. Fluidez de la imagen

Por último, y no por ello menos importante, hay que destacar la limitación del móvil para procesar las imágenes y los algoritmos de detección, de manera que la tasa de FPS sea lo más estable posible.

Tanto la vibración del teléfono (producida por la base en la que se sujeta a la luna delantera del vehículo) como una gran cantidad de movimiento o formas a detectar, hacen que la tasa descienda. Cuanto más vacía esté la imagen, mayor será la tasa.

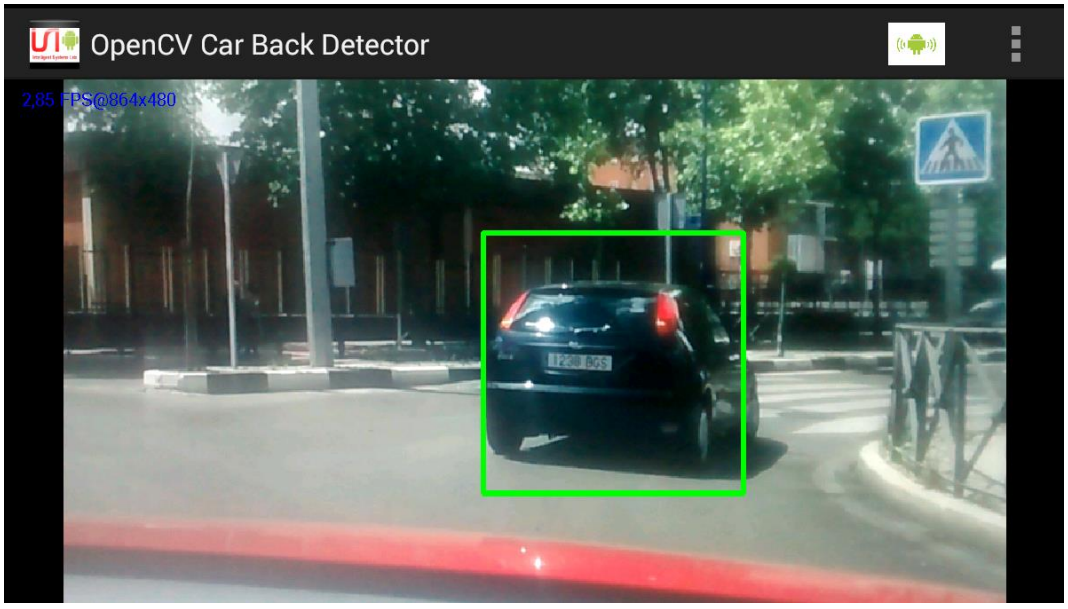

Ilustración 35 – Dentro de coche en movimiento, 2'85 FPS

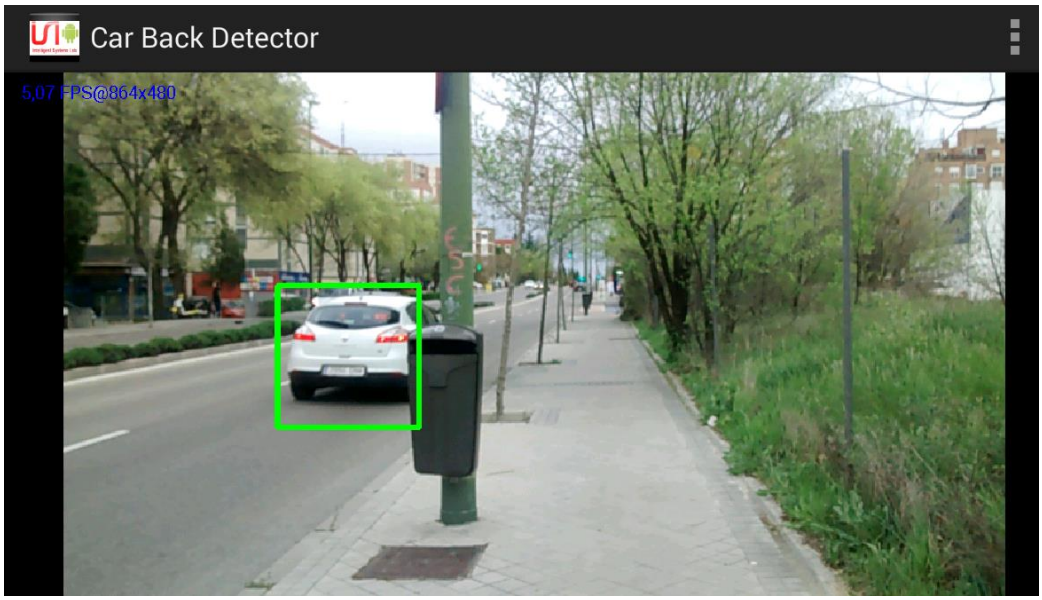

Ilustración 36 - Parado en la acera: 5 FPS

4.5. Características del clasificador

En este punto de la memoria, el lector ya habrá comprendido lo complicado que es obtener un clasificador adecuado para el objetivo perseguido, sea éste la realización de una aplicación de móvil u otro objetivo diferente; a no ser que uno mismo disponga del tiempo (mucho) y los conocimientos para realizar un buen clasificador.

Todo esto implica que las aplicaciones de detección de peatones y vehículos fallen. Los dos clasificadores utilizados fueron facilitados por el tutor. En Internet es prácticamente imposible encontrar clasificadores y los pocos que hay funcionan peor.

En primer lugar, se analizarán los inconvenientes del clasificador de peatones, ligados a todas las limitaciones ya comentadas anteriormente.

- Muchas posturas no son detectadas. Lo son las más comunes: de frente a la cámara, de espaldas, andando, etc.

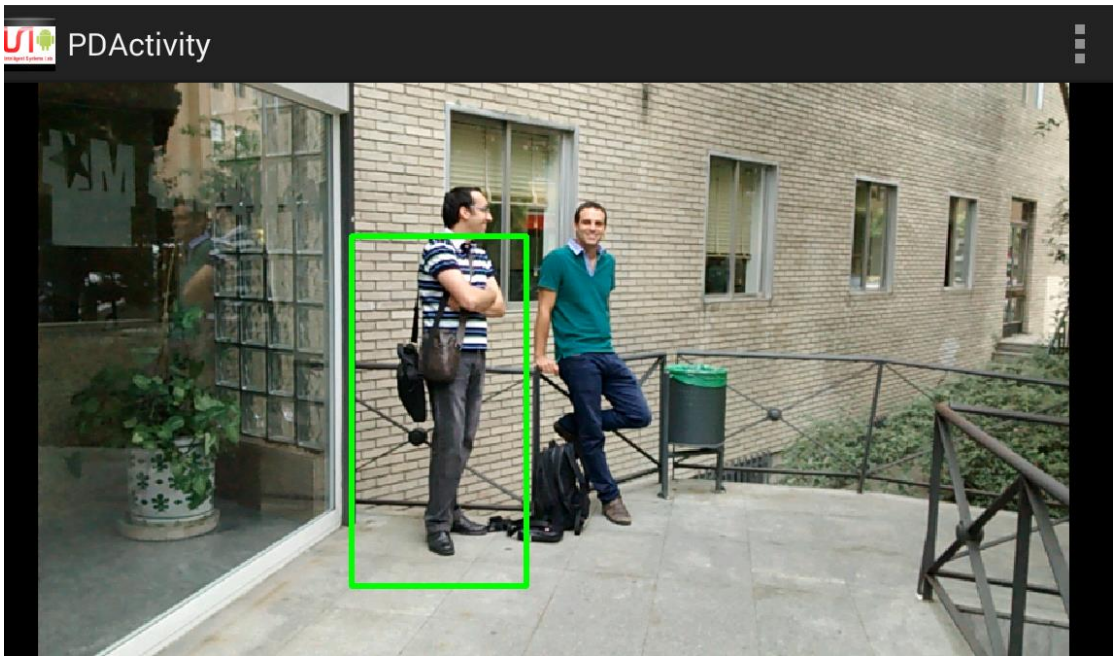

Ilustración 36 - Ejemplo de contraste de posturas.

- Problemas de falsos positivos con objetos de tipo cilíndrico como farolas o papeleras públicas. Tanto el tutor como el jefe de proyecto, expertos en OpenCV, confirmaron que éste es un fallo común en este tipo de clasificadores.

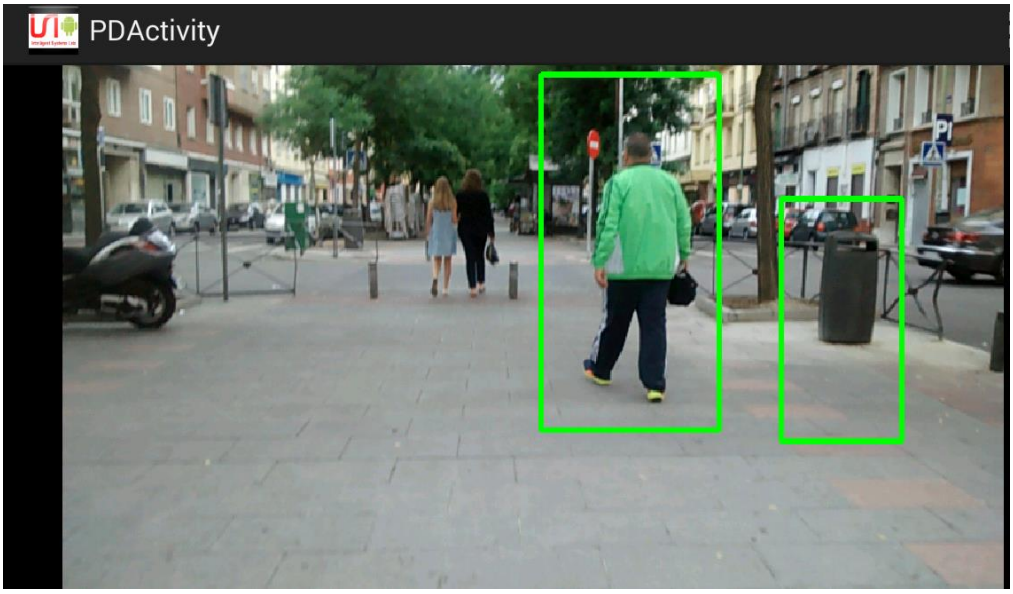

Ilustración 37- Ejemplo de falso positivo

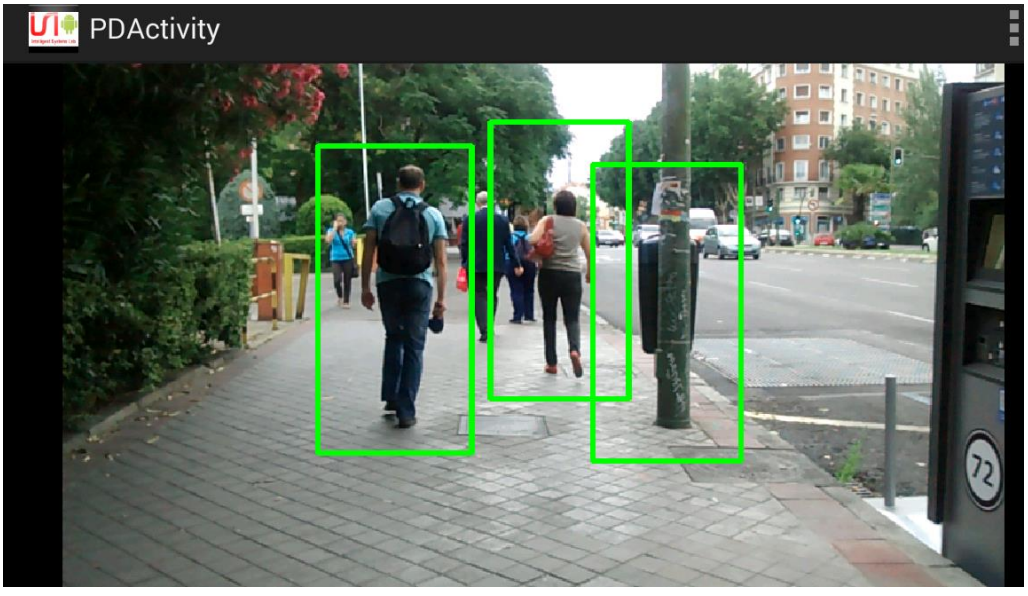

Ilustración 38 - Ejemplo de falso positivo 2

Las ilustraciones 37 y 38 son un buen ejemplo de este problema. Se observa que, al peatón que está a la distancia adecuada, lo detecta de manera perfecta (en movimiento también) pero muestra un falso positivo con la papelera de la derecha. Además, se aprecia cómo los peatones más alejados no son detectados debido a la distancia.

En segundo lugar, se analizarán los inconvenientes del clasificador de vehículos (recuerde que está entrenado, principalmente, con la parte trasera y delantera de los vehículos). Dicho clasificador tiene problemas de detección con:

- Modelos de coches recientes que sigan líneas de diseño modernas.
- Falsos positivos en entornos.

Cabe destacar que, siguiendo a un vehículo introducido en el entrenamiento, se mantiene la detección, siempre que se mantenga una distancia adecuada.

5. PROBLEMAS

Este apartado está enfocado a comentar algunos de los problemas sucedidos durante el desarrollo de este proyecto. Se ha decidido separarlo del apartado de "Limitaciones" puesto que, aunque están relacionados, no son lo mismo.

5.1. Android Studio

Inicialmente, se ideó desarrollar el proyecto utilizando el más reciente entorno de desarrollo Android, AndroidStudio. Durante el primer par de meses del proyecto se trató de continuar con la idea inicial, pero la aparición de numerosos errores, aún no pulidos en la beta del programa, y la falta de información, no ya sobre OpenCV en Android, que también, sino sobre AndroidStudio, imposibilitaron la continuación del proyecto sobre dicho entorno.

Por esto, finalmente se decidió utilizar la SDK de Android para Eclipse, con mucha más información de referencia y mejor preparada para la instalación de las librerías de OpenCV.

5.2. Problemas de framerate

Estos problemas se producen por lo comentado en el apartado "Limitaciones". Evidentemente, se hacen palpables sobre todo a la hora de utilizar la aplicación mientras el usuario se desplaza a alta velocidad.

Tras numerosos intentos, se consiguió mejorar el framerate gracias a cambios en el código.

5.3. Instalación obligatoria de OpenCV

Este mensaje aparece cada vez que el usuario instala la aplicación en un dispositivo que no tiene instaladas las librerías de OpenCV.

Ilustración 39 - Error OpenCV no encontrado

Para solucionarlo (y poder ejecutar la aplicación), es obligatorio descargar de la Play Store la aplicación que las contiene:

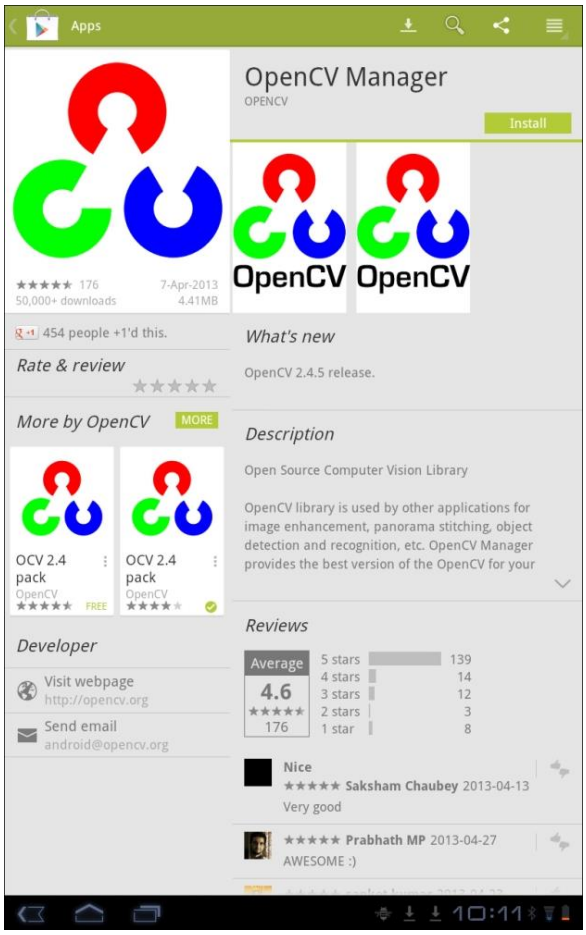

Ilustración 40 - Aplicación librerías OpenCV

5.4. Grabación de pantalla

La mayoría de aplicaciones que existen en Android para grabar la pantalla, muchas de ellas de dudoso origen, requieren que el usuario sea "root" (súper-usuario, con todos los permisos).

Teniendo en cuenta que el teléfono utilizado para desarrollar el proyecto, Motorola Moto G, no estaba rooteado, no ha sido posible utilizar este tipo de aplicaciones.

Por otro lado, existen en la web tutoriales que muestran cómo es posible grabar la pantalla de un dispositivo Android cuya versión sea KitKat (Android 4.4) o superior. Para ello, basta con abrir una consola de Windows e introducir algunos comandos, manteniendo siempre conectado el teléfono al ordenador.

Todos los pasos se graban perfectamente hasta que iniciamos la cámara con las librerías OpenCV. En ese momento la pantalla no "reconoce" que la aplicación funciona en apaisado y la imagen se ve cortada.

Esto imposibilita la correcta grabación del funcionamiento de la aplicación, por tanto se descarta este punto.

Otra opción era utilizar un programa de grabación obtenido en la Play Store, como se comentaba al inicio de este sub-apartado. El HTC Desire utilizado en el desarrollo sí estaba rooteado y por tanto, permitía la instalación de aplicaciones de grabación de pantalla.

Sin embargo, al llegar al momento en el que se ejecuta la cámara con OpenCV, la aplicación dejaba de grabar.

Por todos estos inconvenientes, finalmente se han grabado los vídeos con otro dispositivo apuntando a la pantalla del que ejecuta la aplicación LSIopenCV.

6. TRABAJOS FUTUROS

En este apartado se analizarán los posibles trabajos futuros relacionados con este proyecto. Posiblemente, muchos de ellos puedan tomar este proyecto como base, ejemplo o iniciación en el desarrollo de OpenCV para Android.

6.1. Tracking

Tracking, del inglés seguimiento, es, probablemente, una de las ideas que mejor encajan en la idea de trabajos futuros relacionados con el presente proyecto. Se descartó por no considerarse viable, ya que un algoritmo de seguimiento a nivel Android conllevaría trabajo suficiente para otro proyecto completo, sin tener en cuenta el clasificador.

Se podría tener en cuenta el Filtro de Kalman [22] para el desarrollo de los algoritmos de seguimiento.

6.2. Detección de distancias

Conseguir calcular la distancia a la que se encuentra el objetivo detectado por el dispositivo mediante una serie de cálculos matemáticos o la utilización de sensores.

6.3. Previsión de colisión

Como continuación del proyecto de "Detección de distancias".

Se podría advertir al usuario cuando el objeto detectado se encuentra a una distancia menor de un valor establecido o cuando la distancia se va reduciendo.

6.4. Entrenamiento para móviles

Constituye un proyecto muy interesante para realizar de cara al uso de estas aplicaciones con fines comerciales.

Un entrenamiento estándar, dedicado sólo a móviles, produciría un mayor índice de detección y un número menor de falsos positivos.

Realizar un clasificador conlleva mucho trabajo, como ya se ha explicado en el sub-apartado 9.3. Aunque fuese para móviles, sólo se limitaría a móviles con cámaras similares a la utilizada. Por tanto, no funcionaría igual en un Android de gama baja que en uno de gama alta, si bien lo haría de manera más precisa que el utilizado en este TFG.

No era viable en este proyecto debido al tiempo que implica realizar un clasificador adecuado. Desarrollar un clasificador podría ser un TFG independiente.

6.5. Entrenamiento propio

Similar al apartado anterior, consistiría en crear un clasificador específico para el móvil elegido. Para ello, habría que realizar todas las fotografías del entrenamiento con dicho dispositivo y posteriormente desarrollar el clasificador.

No viable para este proyecto por lo mismo que se ha comentado en el apartado anterior.

6.6. Adelantamientos

Detector de adelantamientos. Calcular la posición de un coche detectado, realizar un seguimiento sobre él y mostrar por pantalla cuándo está realizando un adelantamiento.

6.7. Carriles

Detección de los carriles de una vía. Realizando un análisis de las imágenes, podría realizar cálculos y mediciones de vía: ancho de la vía, curvatura, pendiente, etcétera. De esta forma, alertaría al conductor en diferentes situaciones: vía estrecha, posible salida del carril, etc.

6.8. Señales

Detección de las señales en carretera. Alertaría al conductor cuando apareciesen determinadas señales como un "stop", un ceda al paso, limitación de velocidad, etc.

6.9. Comunicación con ROS

ROS (Robot Operating System) es un meta sistema operativo de desarrollo para robots, como su propio nombre indica. El laboratorio LSI trabaja actualmente con este sistema y podría ser una buena idea establecer una comunicación entre un smartphone y un servidor que contenga ROS, de manera que sea posible enviar datos al SO y trabajar con ellos.

7. CONCLUSIONES

Realizar un TFG de esta envergadura ha supuesto para mí un reto y una gran motivación.

El proyecto tiene como objetivo poner en práctica gran parte de lo aprendido en la carrera pero es también un trabajo que tiene que desarrollar uno por sí mismo. El desconocimiento inicial fue sustituido por la ilusión y las ganas de aprender.

Es cierto que, en este caso, tanto mi tutor como mi director de proyecto podían ayudarme con algunas preguntas, pero no con todas. He ahí uno de los puntos más importantes de este tipo de proyectos: investigar, lograr avances y conseguir que sean entendibles para los demás.

En muchos momentos no sabía cómo avanzar, sobre todo al principio del proyecto, cuando todo fallaba y no había una referencia clara, como pueden ser libros, documentación, etc.

Es para mí un orgullo saber que este proyecto puede servir de referencia a otros compañeros que decidan realizar trabajos similares con el LSI.

Está claro que queda mucho por avanzar en este campo y que esto es sólo un granito de arena en una montaña a la que le queda mucho por crecer. Espero que, en un futuro cercano, sea posible disfrutar de sistemas similares y que dichos sistemas estén al alcance de cualquier ciudadano a un coste mínimo. Quizás en unos años incluso puedan salvar vidas.

Para mí, todo este trabajo ha merecido la pena y agradezco a mi tutor, Fernando, y a mi director de proyecto, Juan, el apoyo y la ayuda que me han brindado en estos siete meses.

8. PRESUPUESTO

En este apartado se describen todos los costes del proyecto, tanto de hardware como de personal.

8.1. Coste de materiales

Los cálculos del coste de materiales del proyecto se han realizado teniendo en cuenta la amortización en el caso de equipos para tratamiento de la información y sistemas y programas informáticos, según los datos del Ministerio de Hacienda [23]para el cálculo de dicha amortización.

Según estos datos, para los bienes informáticos, el coeficiente máximo de amortización sería del 26% y el mínimo del 10% (puesto que el periodo máximo son 10 años). Se ha optado por aplicar el coeficiente máximo.

Duración del proyecto: 7 meses. Febrero 2014 – Septiembre 2014.

Coeficiente de amortización 26%.

Coste de amortización = (Precio del bien *coeficiente de amortización * meses)/12

HARDWARE NECESARIO PARA EL DESARROLLO				
Unidad	**Descripción**	**Precio**	**Coste Amortización**	**Coste proyecto**
PC	Ordenador con Windows 8.1 – SDK Android.	600€	600€*0’26*7/12	91€
Motorola Moto G	Smartphone Android del año 2014.	197’5€	197’5€*0,26*7/12	43’13€
			TOTAL	134’5€

Tabla 5 – Presupuesto

Para el presupuesto del hardware adicional, utilizado en el desarrollo del proyecto pero no imprescindible, se ha establecido un tiempo de tres meses en lugar del total.

HARDWARE ADICIONAL				
Unidad	**Descripción**	**Precio**	**Coste amortización**	**Coste proyecto**
HTC Desire	Smartphone Android del 2010	50€	50€*0'26*3/12	3'25€
Tablet Asus Nexus 7 HD	Tablet Android del 2014.	220€	220€*0'26*3/12	14'3€
Nexus 5	Smartphone Android del año 2014	300€	300€*0'26*3/12	19'5€
			TOTAL	37'05€

Tabla 6 - Presupuesto adicional

8.2. Coste de personal

Para calcular el sueldo se ha tenido en cuenta el "Estudio retributivo del sector TIC español" [24] de Diciembre de 2012, realizado por CONETIC.

Teniendo en cuenta que el salario bruto medio aproximado es de 22.790.10€ al año y que el autor no tiene experiencia, es factible asumir un contrato de prácticas (o trainee en inglés).

En dicho tipo de contrato, durante el primer año, el salario no puede ser menor al 60% del fijado en convenio para un trabajador que desempeñe el mismo tipo de trabajo. Se asumirá un 70% del sueldo, en lugar del mínimo.

Para calcular el sueldo se tendrá en cuenta:

- Duración del contrato: 7 meses.
- Sueldo bruto estimado en base a convenio: 70% de 22.790'10€.
- Cálculo: 22.790'10/12 * 7 *0'7 = 9305'95

CÁLCULO DE SALARIO			
Sueldo bruto anual	**Duración**	**Contrato prácticas**	**Cálculo sueldo total**
22.790'10€	7 meses	70%	9305'95€

Tabla 7 - Costes de personal

8.3. Resumen del presupuesto

En esta tabla se presenta el coste total aproximado para el desarrollo del proyecto:

Descripción	Coste
Costes de materiales	171'55 €
Costes de personal	9305'95 €
TOTAL PARTIDA	**9477'50 €**

Tabla 8 - Resumen del presupuesto total del proyecto

9. NORMATIVA

Licencias del proyecto.

- OpenCV [25]
- Android Open Source Project License. [26]
- Apache License Version 2.0 [27]
- Eclipse [28]
- Android Studio [29]

10. BIBLIOGRAFÍA

[1] Google. Android Development. [Online]. http://developer.android.com/develop/index.html

[2] Unity. (2014) Unity Mobile. [Online]. https://unity3d.com/es/unity/multiplatform/mobile

[3] Itseez. (2014) OPENCV. [Online]. http://opencv.org/

[4] European Commission. EC - Mobility and Transport. [Online]. http://ec.europa.eu/transport/themes/its/index_en.htm

[5] ITS America. ITS America. [Online]. http://www.itsa.org/

[6] ATEC - ITS France. ATEC - ITS France. [Online]. http://www.atec-itsfrance.net/home.cfm

[7] ITS United Kingdom. ITS UK. [Online]. http://www.its-uk.org.uk/

[8] NY State. DOT - NYS. [Online]. https://www.dot.ny.gov/divisions/operating/oom/transportation-systems/systems-optimization-section/ny-moves

[9] Oregon State. ITS - Oregon State. [Online]. http://www.oregon.gov/ODOT/HWY/ITS/Pages/index.aspx

[10] ITS España. ITS España. [Online]. http://www.itsspain.com/itsspain/

[11] Automotive Blog. (2010, Agosto) Youtube. [Online]. https://www.youtube.com/watch?v=w2pwxv8rFkU

[12] Volvo. (2012, Oct.) YouTube. [Online]. https://www.youtube.com/watch?v=XBQ4NIJtbX0

[13] Volvo. (2013, Mar.) Web de Volvo. [Online].

http://www.volvocars.com/es/top/about/news-events/pages/default.aspx?itemid=108

[14] Mercedes. Web de EURONCAP. [Online]. http://es.euroncap.com/es/rewards/mercedes_benz_pre_safe.aspx

[15] Toyota. (2014, June) Consumer Reports. [Online]. http://www.consumerreports.org/cro/news/2014/06/toyota-s-pedestrian-detection-system-makes-an-impact-at-the-track/index.htm

[16] Fundación Eduardo Barreiros. (2014, Julio) [Online]. http://www.fundacionbarreiros.com/es/fundacion/noticia/148

[17] LSI-UC3M. (2014, Mayo) Youtube. [Online]. https://www.youtube.com/watch?v=ztPm4WDt-6c

[18] LSI-UC3M. (2014) UC3M. [Online]. http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/lab_sist_inteligentes/repository

[19] LSI-UC3M. UC3M. [Online]. http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/lab_sist_inteligentes/publications

[20] Google. (2014) Android Developers. [Online]. https://developer.android.com/about/versions/kitkat.html

[21] OpenCV. (2014) Samples OpenCV. [Online]. http://opencv.org/platforms/android/opencv4android-samples.html

[22] UC3M. (2004) TSC-UC3M. [Online]. http://www.tsc.uc3m.es/~mlazaro/Docencia/Doctorado/FiltAdapt/Kalman.pdf

[23] Ministerio de Hacienda. (2014, Mes) Agencia Tributaria. [Online]. http://www.agenciatributaria.es/AEAT.internet/Inicio_es_ES/_Segmentos_/Empresas_y_profesionales/Empresarios_individuales_y_profesionales/Rendimien

tos_de_actividades_economicas_en_el_IRPF/Regimenes_para_determinar_el_rendimiento_de_las_actividades_economic

[24] CONETIC. (2012, Diciembre) http://www.aeiciberseguridad.es/. [Online]. http://www.aeiciberseguridad.es/descargas/categoria6/8774932.pdf

[25] OpenCV. (2014) OpenCV License. [Online]. http://opencv.org/license.html

[26] Android. (2014) Android OSP License. [Online]. https://source.android.com/source/licenses.html

[27] The Apache Software Foundation. Apache License, v2.0. [Online]. http://www.apache.org/licenses/LICENSE-2.0

[28] The Eclipse Foundation. (2014) Eclipse Public License. [Online]. https://www.eclipse.org/legal/epl-v10.html

[29] CreativeCommons. CC. [Online]. http://creativecommons.org/licenses/by/2.5/

[30] OpenCV Developers Team. Web de OpenCV. [Online]. http://opencv.org/

[31] Willow Garage, Itseez Intel. OpenCV Documentation. [Online]. http://opencv.org/documentation.html

[32] Willow Garage. Willow Garage. [Online]. https://www.willowgarage.com

[33] Itseez. Itseez.com. [Online]. http://itseez.com/

[34] Fellow, IEEE, and Tim Bailey Hugh Durrant-Whyte. Berkeley. [Online]. http://www.cs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf

[35] Søren Riisgaard and Morten Rufus Blas. MIT. [Online]. http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf

[36] Google. Android Developers. [Online].
http://developer.android.com/index.html

ANEXO I – OPENCV EN ANDROID

Este primer anexo es un tutorial sobre cómo configurar las librerías OpenCV en la SDK de Android para Eclipse.

Autor: Alejandro Ramos López

Ilustración 41 - UC3M - LSI

Tutorial para acceder a la cámara de una aplicación Android y empezar a programar incorporando las librerías de OpenCV.

Primero, es necesario descargar Eclipse con los "plugins" de desarrollo de Android. A través de este enlace se puede descargar el SDK (Software Development Kit) de Eclipse para Android.

http://developer.android.com/sdk/index.html

Para probar la aplicación OpenCV es necesario un smartphone con una versión de Android 2.2 (Froyo) o superior.

Se requiere al menos una cámara. Si tiene cámara frontal, mejor.

Otros aspectos a destacar son el autoenfoque y el acceso a la Play Store de Google.

No se recomienda utilizar el emulador ya que algunas de las instrucciones utilizadas por OpenCV en el acceso a la cámara trabajan a bajo nivel y pueden causar **errores** en las cámaras virtualizadas.

Si no quedase más remedio, se puede configurar el AVD (Android Virtual Device) para que utilice como cámara la webcam del ordenador.

En segundo lugar, se ha de descargar la librería de OpenCV con sus ejemplos.

http://sourceforge.net/projects/opencvlibrary/files/opencv-android/

Durante la redacción de este tutorial, la versión de OpenCV es la 2.4.8. Simplemente, se debe seleccionar la última disponible.

Se importa la librería del archivo que se ha descargado, **OpenCVlibrary**. Se encuentra en la carpeta "sdk" del archivo descargado.

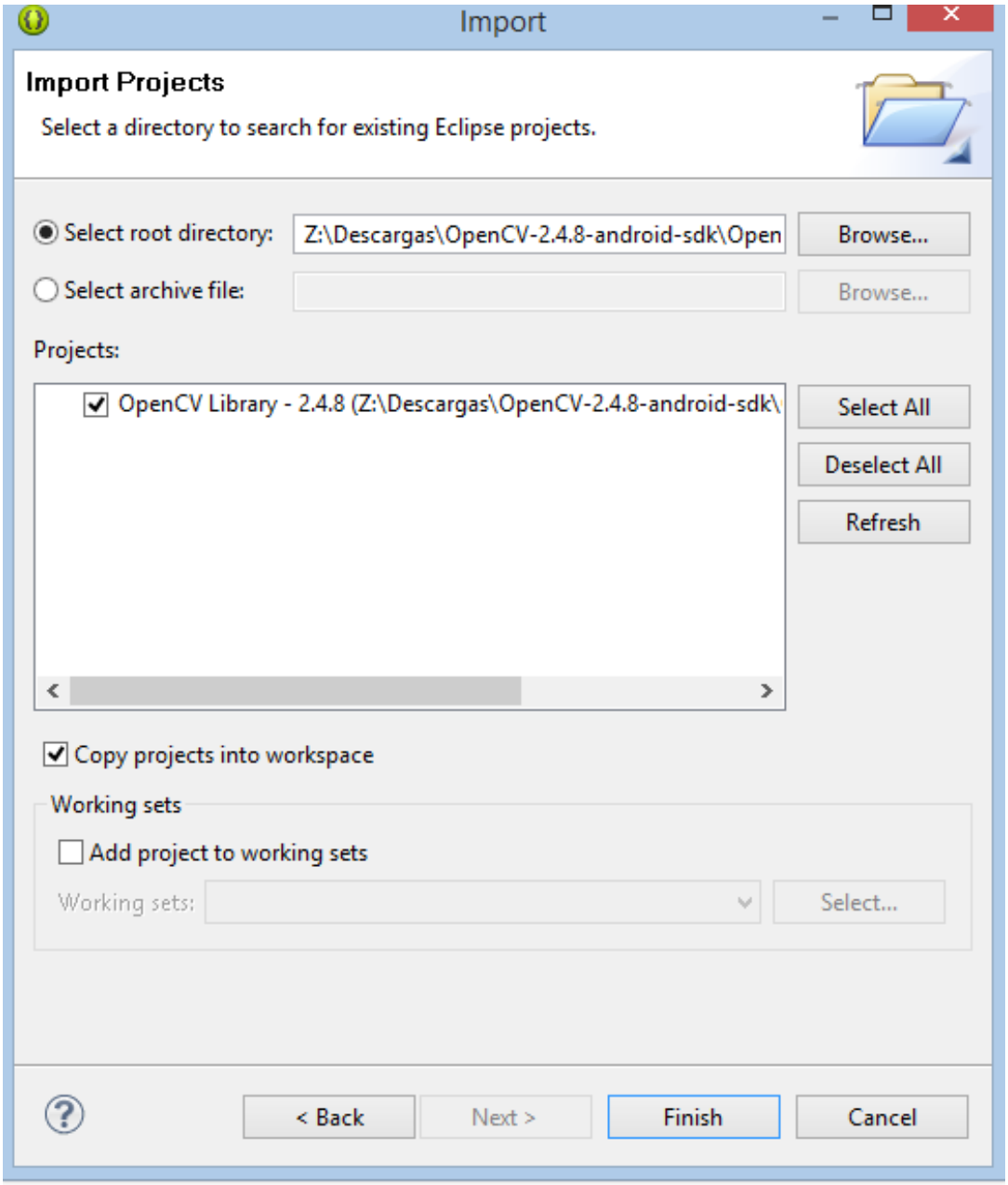

Ilustración 42– Importación de las librerías OpenCV

Si se desea, también se puede importar los ejemplos:

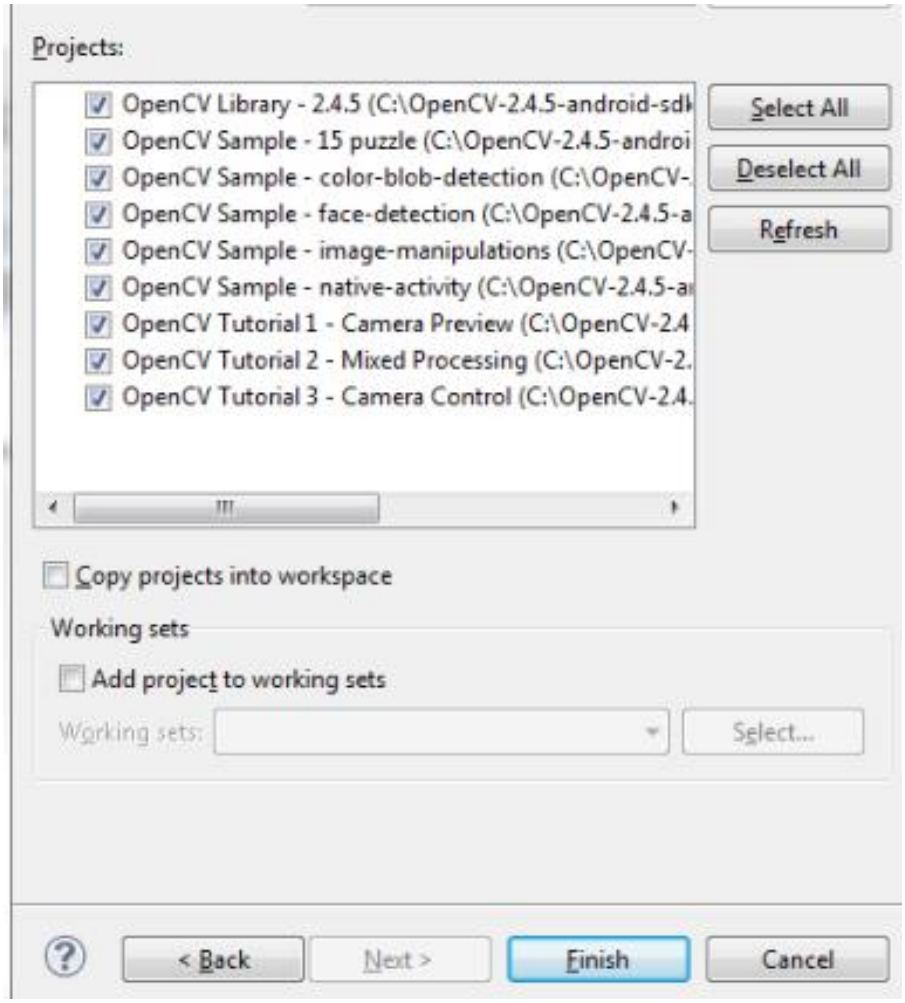

Ilustración 43 - Importación de los ejemplos

Al importar, tanto la **librería** como los ejemplos, **aparecerán errores.** Esto es debido a la versión de Android. En la siguiente imagen se aprecia un ejemplo de los errores:

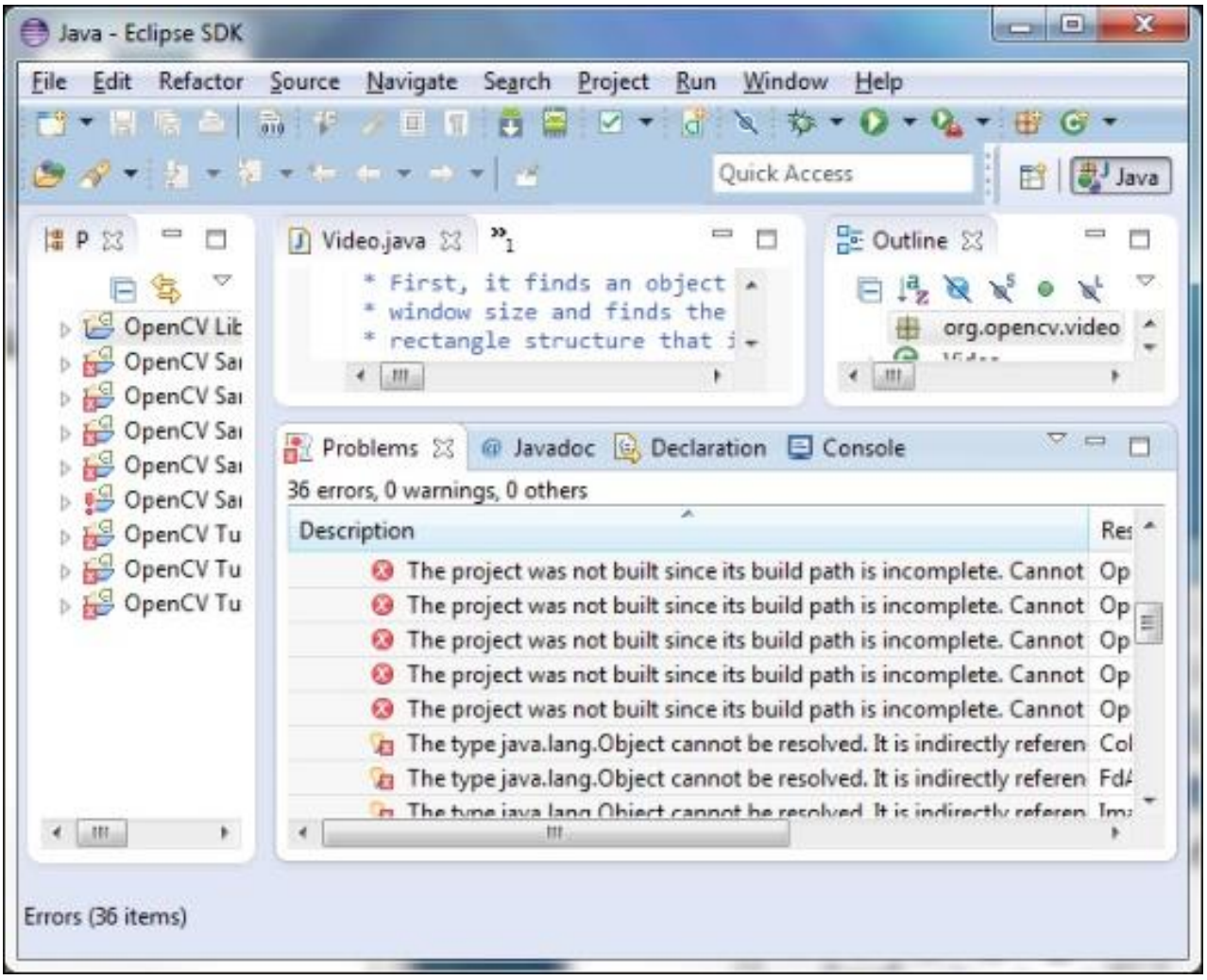

Ilustración 44 - Errores iniciales

Para solucionarlo, pulsar botón derecho sobre la librería: "Properties>Android> Project Build Target > Versión de Android" sobre la que se esté desarrollando el proyecto.

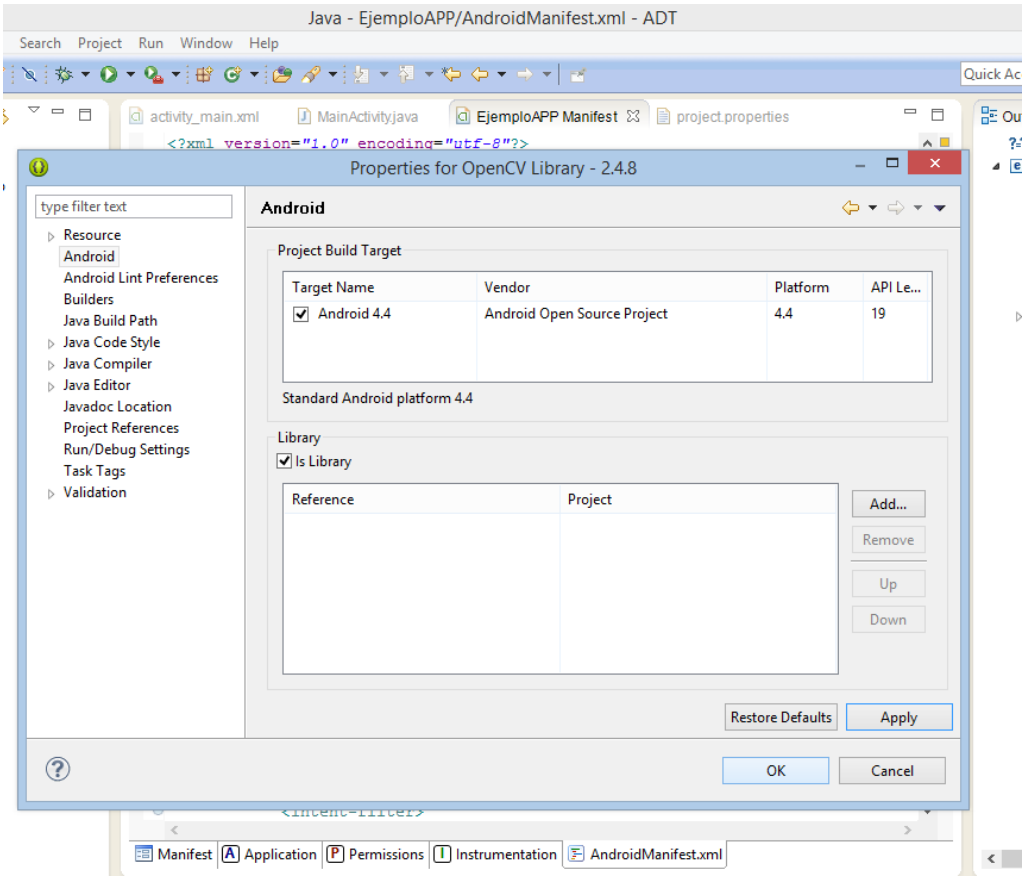

Ilustración 45 - Modificación de la versión de Android

Para los ejemplos habría que hacer lo mismo.

Con esto estaría solucionado pero, **si estás desarrollando con Mac o Linux**, puede darse otro error similar a éste:

Program "{ndk}/ndk-build.cmd" not found in PATH

En este caso, se pulsa botón derecho: "Properties>C/C++ Build". En este apartado se debe editar el contenido de "Buildcommand", borrando el texto ".cmd", de manera que quede así:

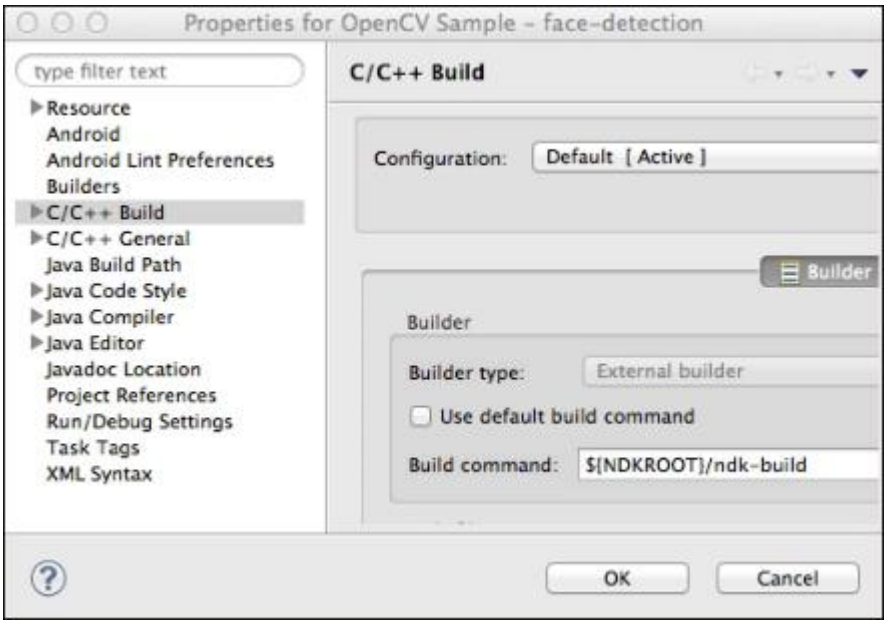

Ilustración 46 - Corrección en la NDK

Una vez corregidos estos errores, para incluir la librería de OpenCV en el proyecto hay que importarla. Clic al botón derecho >Properties>Android> Library >Add.

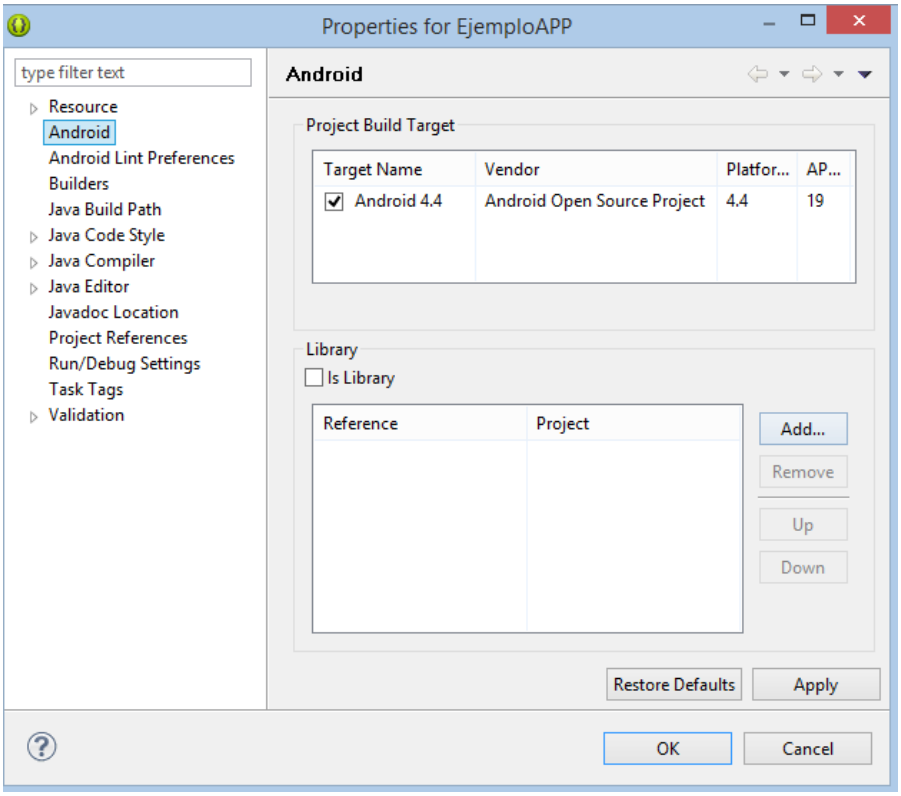

Ilustración 47 - Añadiendo la biblioteca

Y se añaden las librerías de OpenCV.

A la hora de **ejecutar** cualquier **aplicación OpenCV por primera vez,** el teléfono avisará de que no están instaladas las librerías OpenCV. **Es necesaria la "app" de OpenCV**. Para ello, se debe descargar de Google Play Store.

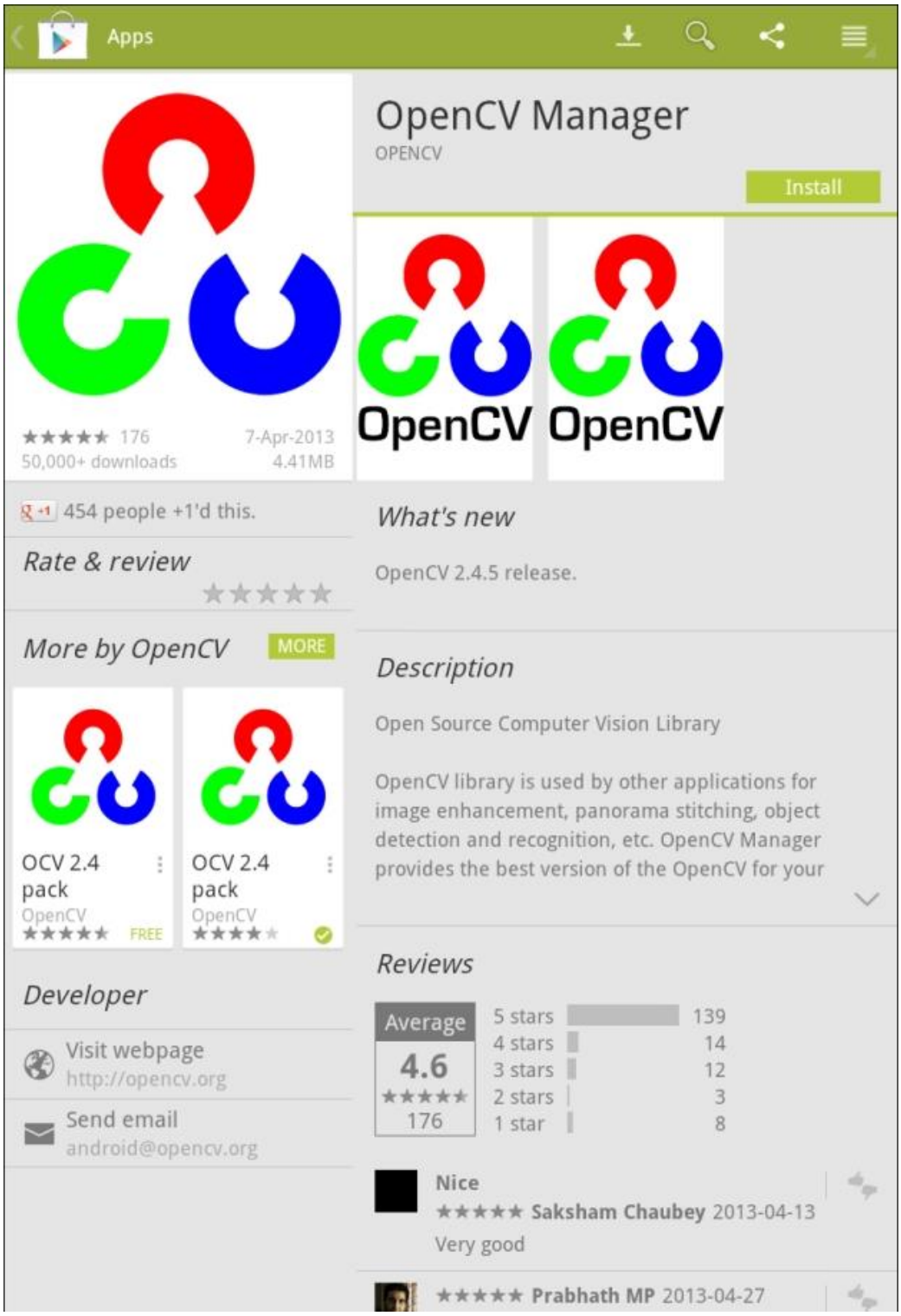

Ilustración 48 - Aplicación en el App Store

Si no tiene acceso al Store desde el smartphone, deberá instalarla a través del USB.

Por último, para poder utilizar la cámara en la aplicación será necesario añadir el **permiso de acceso a la cámara** en el "**AndroidManifest.xml**" de la aplicación, en el cual se definen todos los permisos de la "app", sus "activitys", la versión de Android que se utiliza, los paquetes de código Java, etc.

La línea a añadir es la siguiente:

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Ilustración 49 - Añadiendo permiso de cámara

Además del permiso de utilización de la cámara, para utilizar OpenCV es conveniente añadir las siguientes características para el manejo de la cámara:

```
    <uses-
featureandroid:name="android.hardware.camera"android:required="false"/
>
    <uses-
featureandroid:name="android.hardware.camera.autofocus"android:require
d="false"/>
    <uses-
featureandroid:name="android.hardware.camera.front"android:required="f
alse"/>
    <uses-
featureandroid:name="android.hardware.camera.front.autofocus"android:r
equired="false"/>
```

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.lsi.lsiappdet"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk
8          android:minSdkVersion="15"
9          android:targetSdkVersion="17" />
10
11     <uses-permission android:name="android.permission.CAMERA" />
12
13     <uses-feature
14         android:name="android.hardware.camera"
15         android:required="false" />
16     <uses-feature
17         android:name="android.hardware.camera.autofocus"
18         android:required="false" />
19     <uses-feature
20         android:name="android.hardware.camera.front"
21         android:required="false" />
22     <uses-feature
23         android:name="android.hardware.camera.front.autofocus"
24         android:required="false" />
25
```

Ilustración 50 - Añadiendo permiso de cámara

Ahora bien, una vez está todo preparado para utilizar OpenCV, **se creará una nueva "activity"** de Android para trabajar con OpenCV.

En el layout.xml de la "activity" de OpenCV se introduce lo siguiente:

```
    <LinearLayoutxmlns:android="http://schemas.android.com/apk/res/andr
oid"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:opencv="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <org.opencv.android.JavaCameraView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:visibility="gone"
    android:id="@+id/HelloOpenCvView"
    opencv:show_fps="true"
    opencv:camera_id="any"/>

    </LinearLayout>
```

Aparte del "ítem" configurador de la cámara, no olvidar introducir la referencia a OpenCV (señalada en negrita) en el primer "LinearLayout", ya que si no dará error.

Para utilizar la "activity" de la cámara, se debe realizar una transición con un "Intent" de Android desde la "activity" inicial o desde la que se quiera pasar a utilizar la cámara.

```
    publicvoidgoToCamera(View v) {
    if (v.getId() == R.id.startButton) {
      Intent i = newIntent(MainActivity.this, OpenCVactivity.class);
            startActivity(i);

              }
```

En este caso, simplemente es una función que se activa si se pulsa el botón de Inicio, al cual he llamado "startButton". La idea es que, al pulsar el botón, se pase a la "activity" (OpenCVactivity) que trabaja con OpenCV a través de la cámara.

Este ejemplo está declarado como se puede ver en la clase MainActivity, ya que la intención es pasar a utilizar la cámara desde la pantalla inicial.

Una vez se consigue activar la cámara, ya es posible empezar a modificar la clase que se ha creado para trabajar con OpenCV.

Enlaces de interés:

http://developer.android.com/sdk/index.html

http://opencv.org/platforms/android.html

http://www.intorobotics.com/tutorials-setup-opencv-for-android-ios-linux-and-windows/

http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html

Elaborado por:

Alejandro Ramos López, alumno del Grado en Ingeniería Informática en la UC3M.

alexrl1990@gmail.com

alexrl1990@hotmail.com

100080835@alumnos.uc3m.es

ANEXO II – APP BASE

En este segundo anexo de la memoria se explica cómo **añadir nuevos proyectos** a la aplicación base desarrollada por el alumno Alejandro Ramos.

En primer lugar, se ha de importar el código de la aplicación base al entorno de desarrollo.

Una vez importado el código base y con las **librerías OpenCV** correctamente dispuestas en el entorno de desarrollo, se han de tener en cuenta **dos alternativas** a la hora de añadir nuevas "subaplicaciones" a la aplicación OpenCV del LSI. Dichas alternativas serán las siguientes:

- **Añadir aplicaciones a la lista del LSI.**
- **Añadir un nuevo botón de aplicación.**

a) Añadir aplicaciones a la lista del LSI

En este caso, simplemente se tienen que añadir la aplicación deseada a la lista de "apps" que ya muestra el botón "Ejecutar aplicación", dentro de la interfaz "Aplicaciones LSI".

Ilustración 51 - Interfaz inicial

Ilustración 52 - Aplicaciones LSI

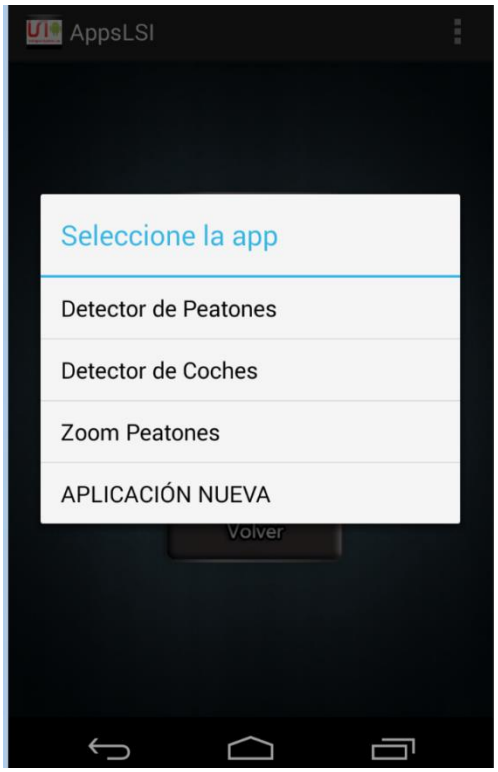

Ilustración 53 - Aplicación nueva

Para que el botón de la nueva aplicación aparezca en pantalla, ilustración 53, es necesario **añadirlo** en orden al **array** de **String** "types", como se puede ver en la siguiente ilustración. Además, es imprescindible añadir un nuevo "case" en el switch, a través del cual se indica la activity hacia la que se dirigirá el nuevo botón.

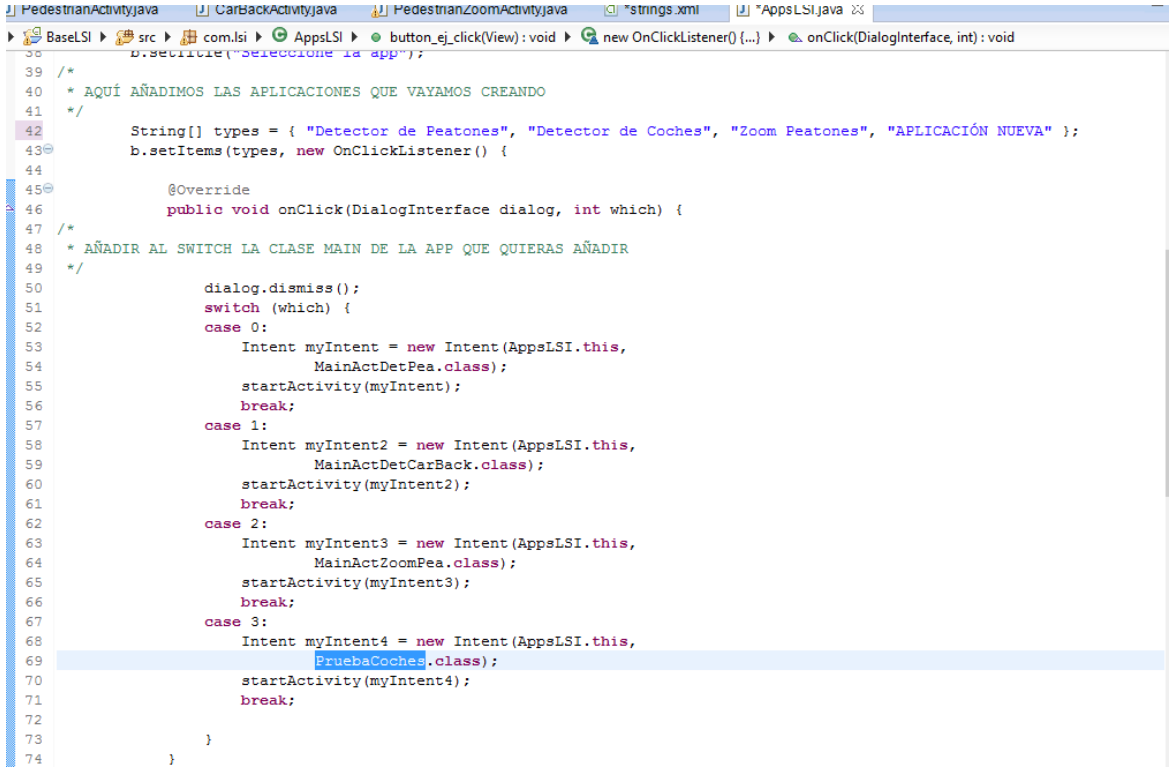

Ilustración 54 - Añadiendo aplicación nueva

En este caso, la aplicación de ejemplo para el tutorial tendría como título "APLICACIÓN NUEVA" en el objeto AlertDialog. Al pinchar en el botón, el programa se dirigirá a la activity "PruebaCoches".

Depende del desarrollador si la clase a la que se transita es ya directamente una ejecución de la cámara utilizando OpenCV o si es una interfaz de acceso a otras activities.

b) Añadir un nuevo botón de aplicación

Para este apartado es **recomendable** disponer de **Adobe Photoshop**, si se quiere conseguir un **resultado idéntico** al disponible inicialmente.

En primer lugar, se ha de abrir el archivo "**botonParaModificar.psd**" con el cual se podrá realizar el nuevo botón para la aplicación.

Para el botón en estado normal, sin presionar, basta con modificar el texto con lo que se le quiera añadir. Una vez se le haya añadido el texto, es **importante** guardarlo con la opción "**Guardar para web**".

Para el botón presionado se han de seguir los siguientes pasos:

- Mismo texto del botón sin presionar.
- **Combinar capas**.
- Modificar **tono/saturación**, **colorear** con los siguientes valores:
  - **Tono**: 235
  - **Saturación**: 20
  - **Luminosidad**: 0
- Tamaño inicial de letra: 24 pt.
- Fuente: Microsoft Tai Le.
- Los efectos de letra, trazo y sombra paralela están ya configurados, pero se pueden modificar.
- Se puede mover ligeramente el texto hacia arriba a la derecha del texto, así aumenta la sensación de presión del botón.

Después, se deben añadir los botones a las carpetas de "res>drawable" correspondientes, en función del tamaño.

Una vez importados los botones, será posible añadirlos ya al fichero XML del activity correspondiente. En este caso de ejemplo, se crearía un nuevo "relativelayout", con las mismas características de los ya existentes, de manera que se repartan los botones de manera equidistante por toda la interfaz. Si bien esta es la posibilidad de ejemplo, el desarrollador puede decidir la estructura de la activity según le interese.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1" >

    <ImageButton
        android:id="@+id/startButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:background="@android:color/transparent"
        android:contentDescription="@string/startApp"
        android:focusable="true"
        android:onClick="button_click"
        android:src="@drawable/bot_ap" />
</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1" >

    <ImageButton
        android:id="@+id/ejemplosButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:background="@android:color/transparent"
        android:contentDescription="@string/ejemplos"
        android:focusable="false"
        android:gravity="center_vertical"
        android:onClick="button_click"
        android:src="@drawable/bot_ejemplos" />
</RelativeLayout>
```

Ilustración 55 - Ejemplo declaración botón XML

Además, se ha de destacar la función "onClick" añadida a cada botón. En este caso redirige a la función "button_click" de la activity principal de la aplicación LSIopenCV. Dicha función contiene un "switch" que redirige a la activity deseada según el botón que se pulse.